- Use the built-in feature `Export public pages`.
  - Chose a **separate folder** (`$web/`) to checkout the `gh-pages` branch, then export to that folder.
    - Warning: Don't export to the same folder with the `main` branch (`$src/`), because their contents are very different.
    - Manually copy the whole `assets` folder from `$src/` to `$web/`
      - because [!] The built-in publisher still cannot handle assets (embedded or linked) other than embedded image.
  - The folder `$web/` can be launched by Web Servers.
  - When `gh-pages` branch is pushed to GitHub, GitHub Pages will deploy it to `https://$user.github.io/$repo/`.
  - [!] The built-in publisher does not give option to set `theme-mode: light/dark` and `accent-color: blue` and it always use the default theme (`data-color=logseq`).
  - So i customize `data-color=logseq` to match CreatZy theme.
  - [!] The built-in publisher still cannot handle assets (embedded or linked) other than embedded image.
    - ⇒ Manually copy the whole `assets` folder from `$src/` to `$web/`
- CANCELLED Use publish-SPA GitHub Action to publish to GitHub Pages.
  - [!] The published SPA has some style mismatch with the desktop app.
  - [!] The published SPA still cannot handle assets (embedded or linked) other than embedded image.
  - CANCELLED Local publishing with publish-SPA CLI to publish local graph of HTV's works.
    - `publish-spa` requires `logseq` to be built for it to use `logseq/static`, but `logseq > yarn install` fails :(

```
yarn install > Link step > logseq@workspace:. must be built because it never has been before

Ambiguous Syntax Error: Cannot find which to pick amongst the following alternatives:
  0. yarn install [--json] [--immutable] [--immutable-cache] [--check-cache] [--inline-builds] [--mode #0
  1. yarn install [--json] [--immutable] [--immutable-cache] [--check-cache] [--inline-builds] [--mode #0
While running --cwd tldraw install
```

- CANCELLED Publish linear (long-form) docs with Hugo publish (logseq-schrodinger) for publishing to Hugo.
- Because the exported SPA is too large (~90MB including Electron), it's better to write a Custom Logseq publish via Web API.

Manually publish with Block copy and JavaScript or vim commands

draft note → published note → published Markdown → HTML → PDF

- First, copy the block to be published to a page in `pages/publish/`, e.g. [[Logseq publish]].
  - Copy to external text editor to get text of block refs.
  - Copy `{{embed}}`ed contents
  - Remove additional notes & tasks
  - [!] The biggest problem is block refs
    - The internal refs to blocks within the page should be resolved automatically.
    - The external refs should be resolved to plain texts.

Then, load the published note to converter script

- publish Markdown
  - convert newline `\n` to `<br />` (skipped)
    - Note: newline is meaningful in Logseq but not in the standard Markdown (line break requires two spaces at line end or a backslash, for the historical plain text email).
      - empty line is treated as paragraph break `</p><p>`, not line break `<br />`.
    - Skipped: The detection of wich lines to be converted is complicated, so we use the supported `markdownit({breaks: true})`
      - For standard markdown, line breaks should be inserted manually when needed.
  - convert metadata to `<a id="UUID" data-property="..." data-logbook="..."></a>`
  - `looseList` to avoiding the style inconsistency between "loose lists" and "tight lists".
    - Issue: Any blank line in any list item makes that list "loose", i.e., all other items in the same list (at the same level) will be wrapped in `<p>` which will be rendered different (more line space) from the unwrapped items of the default "tight list".
    - [!] Cannot make soliton lists "loose" simply with blank line!
      - soliton list: only one item with no sub-list
      - ~~Trick: Add another line containing ` ` after the blank line.~~
        - This line has no effect with normal text, but adds new code block after code block!
      - ⇒ Hack: set all `markdown-it`'s tokens with `"hidden": false`.
        - break down `mdi.render()` into:
          Markdown →`mdi.parse()` → token stream →`mdi.renderer.render()` → HTML
          with `mdi = window.markdownit() || new MarkdownIt()`
          - Ref source code: MarkdownIt.prototype.render

- `markdown-it` should have a rule to disable “tight list” feature.

  - Some conventions, like blankLineBeforeCodeBlock, require loose list.
  - The `<p>` wrapping is also convenient for `<div slot="unfoldable">` in `<folder-div>`.
  - References:
    - CommonMark talk: Why are there even tight lists?
    - Markdown-It issues: #728, #678,
    - Prettier's issue
    - Djot's issue: Revisit the concept of tight/loose list.
  - We also use style `li p {margin}` to fix this display issue.
- `flattenHeadings` to unitemize headings & remove first tabs
- process block link/ref -> `#` anchor link
  - Detect unresolved links
  - Convert `((block ref))` and empty link `[](((UUID)) "comment")` to `[target block title](((UUID)))`
    - All links & block refs in the `target block title` is replaced by **plain text** due to the limitation of HTML anchor.
    - Use `mapUuid[id] = blockTitle` to track block titles; add `target block title` to empty block ref/links in the topo-order of block ref dependency
  - Note: Link text can contain **matched** square brackets `[]`, and external link target can contain **matched** parentheses `()` ... up to **3 levels**.
    - Some Markdown implementations do allow ***matched*** *brackets/parentheses* in link.
      - E.g. This links to [wiki] Parenthesis_(rhetoric)
      - Supported by Logseq, Markdown-It, GitLab
      - Unsupported by GitHub, Markdown Viewer (Browser Extension), Stack Exchange
      - Parentheses in link target is supported by most of them: HitHub, Markdown Viewer, Stack Exchange,
- Regex pattern for `n`-level-nesting balanced brackets:
  - For unbound `n`, recursive regex is required.
  - For specific `n` (= 3 in our case), we can construct the pattern with this algorithm:

```
/**
 Examples:
    [A [very [very [very]...] messy] link](http://to(some(weird(...))).href "with link tip")
    // 3 levels of []
    patText = balancedBracketsRegexPattern('[',']','',3,true)
    // 3 levels of () and exclude space & quote of the link tip
    patHref = balancedBracketsRegexPattern('(',')',' "',3,true)
*/
function balancedBracketsRegexPattern(open='[', close=']', excludes='', depth=1, unrolled=false
{
    let lo = '\\'+open, lc = '\\'+close;  // literals
    let noBracket = '[^'+lo+lc+excludes+']';
    // Pattern variants:
    let t = unrolled ? 1 : 0;
    let p = [ // [open, close]
        [// simple pattern
            lo
            + '(?:'+ noBracket  +'|'/*inner level*/,
              ')*' +
            lc
        ],
        [// unrolled pattern for efficiency
            lo +
            noBracket+'*'
            + '(?:' /*inner level*/,
            noBracket+'*'
            + ')*' +
            lc
        ]
    ];

    // Generate the pattern
    let innermostPair = lo + noBracket+'*' + lc;
    let openBrackets  = p[t][0].repeat(depth);
    let closeBrackets = p[t][1].repeat(depth);

    // Return the pattern
    pattern = new RegExp(openBrackets + innermostPair + closeBrackets);
    return pattern;
}
```

  - Ref: Regular expression to match balanced parentheses
- Idealy, link text should not contain brackets, and parentheses in link target should be esceped: `(` = `%28`, `)` = `%29`
  - E.g. This links to {wiki} Parenthesis_(rhetoric)
  - Because any unmatched bracket/parenthesis will break the link sysntax with broken text displayed.

    - This has [only close braket]](undefined)
    - This has (only open parenthesis](https://en.wikipedia.org/wiki/Parenthesis_(rhetoric))
    - This has only close parenthesis) ...
- TODO for `logseq.order-list-type:: number`, replace items bullets with numbers
- process code blocks
    - `blankLineBeforeCodeBlock` (⇒ looseList) for strict conventions like in GitLab
    - unitemize items with code blocks only.
    - replace the double space with `\t`.
- warn for external links to relative paths: non-HTTP
- replace straight quotes `"..."` with curly quotes `“...”` (and `'...'` with `‘...’`)
    - ⇐ ~~Auto-complete &~~ typing assistant for **quotation marks**, symbols...
    - This has been done by Markdown-It's `typographer: true` in smartquotes.mjs.
    - We implement our own function `replaceQuotes(ln)` to handle our markdown source in Logseq.
    - Multi-line quotes are not supported.
    - Test text:

```
This "quotation" is for sth like "quot-"+"-ation" or sth like "12398 ^724_242!?" or "- abc =", but n
This 'quotation' is for sth like 'quot-'+'-ation' or sth like '12398 ^724_242!?' or '- abc =', but n
"line start" and "line end"
'line start' and 'line end'
American style: "double quotes contain 'single quotes'"
British style:  'single quotes contain "double quotes"'
```

    - TODO Process markers like `TODO`, `CANCELLED`, ...
- markdown → HTML: using markdown-it
    - `MarkdownIt.options`
        - `html: true, breaks: true` for comatible with Logseq
        - `typographer: false`: we do `replaceQuotes()` ourselves.
    - make item lists foldable with custom element `<folder-div>`
        - For easy DOM traversal, set looseList = true & flattenHeadings = false.
            - For block title in `<div slot="unfoldable">`: Use `looseList` to wrap all item contents into `<p>`, then use `node.children[0]` to access its first child to get block title, instead of traversing `node.childNodes[]`.
            - Note that, even with `looseList`, the `<li>` always contains meaningless newline-only text nodes which will appear in `node.childNodes[]`.
        - headings in `<div slot="unfoldable">` are automatically detected and moved to `<div slot="heading">`
- HTML to PDF conversion
    - TODO programmatically print HTML page to PDF using Chrome.
        - The printed PDF lacks bookmarks, used for simple page view only.
    - **layout** with bookmarks using DocRaptor
        - Pagination for PDF: Summary & Details
            - Summary: each headings has a link (whole text or `...`) to the item in Details
        - Reduce list indent
            - Browser default: `padding-inline-start: 40px;`
            - Prince default: `margin-left: 52px;`
            - ⇒ reset `padding-inline-start: 0px;` and set `margin-left: 20px;`
- [!] JSON `fetch()` error
    - Request with `mode: 'no-cors'` ⇒ opaque response with `status = 0` & `ok = false`.
    - Serve page with non-secure `HTTP` protocol ⇒ `401 - Unauthorized`
    - Calling `fetch()` with either header `Authorization` or `Content-Type: application/json` ⇒ CORS preflight `OPTIONS` request is sent by browser. (ref: StackOverflow)
    - Calling `fetch()` with `content-type: application/json` ⇒ `Status Code: 400 Bad Request`
        - The preflight request is OK, though!?!
            - Request `OPTIONS` → `200 OK`

```
Request URL: https://api.docraptor.com/docs
Request Method: OPTIONS
Status Code: 200 OK
Remote Address: 34.226.73.93:443
Referrer Policy: strict-origin-when-cross-origin
access-control-request-headers: authorization,content-type
access-control-request-method: POST
origin: https://myip
priority: u=1, i
referer: https://myip/
sec-fetch-dest: empty
sec-fetch-mode: cors
```

```
sec-fetch-site: cross-site
```

- Response

```
access-control-allow-headers: authorization,content-type
access-control-allow-methods: GET,PUT,POST
access-control-allow-origin: https://myip
access-control-max-age: 900
cache-control: no-cache
referrer-policy: strict-origin-when-cross-origin
server: nginx
x-frame-options: SAMEORIGIN
x-permitted-cross-domain-policies: none
```

- But the real POST request is NG!?!
  - CORS Error

```
Access to fetch at 'https://api.docraptor.com/docs' from origin 'https://myip' has been bloc
```

  - Request `POST` → `400 Bad Request`

```
Request URL: https://api.docraptor.com/docs
Request Method: POST
Status Code: 400 Bad Request
Referrer Policy: strict-origin-when-cross-origin
accept: */*
accept-encoding: gzip, deflate, br, zstd
accept-language: en-US,en;q=0.9,vi-VN;q=0.8,vi;q=0.7,ja-JP;q=0.6,ja;q=0.5
authorization: Basic bTdEaHJuX0FzZXpWOTRDM1ZMLUI6
content-length: 15
content-type: application/json
origin: https://myip
priority: u=1, i
referer: https://myip/
sec-ch-ua: "Chromium";v="128", "Not;A=Brand";v="24", "Google Chrome";v="128"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
sec-fetch-dest: empty
sec-fetch-mode: cors
sec-fetch-site: cross-site
user-agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/12
```

  - Response with **no** `Access-Control-Allow-Origin` header

```
cache-control: no-cache
content-length: 149
content-type: application/xml
date: Wed, 30 Oct 2024 11:56:55 GMT
server: nginx
set-cookie: eb_tracking_id=27f076a0-1782-4eab-b2db-82e421aff0a8; domain=.docraptor.com; path=
strict-transport-security: max-age=63072000; includeSubDomains
x-request-id: 2cbb044e-5120-4b63-990d-5373c8135f23
x-runtime: 0.006626
```

- The official docraptor-1.0.0.js use `form.submit()` instead of JSON, with `user_credentials` instead of `Authorization: 'Basic API-key:'`
  - while its docs says

    > HTTP Basic Authentication (preferred [over Query Parameter Authentication `user_credentials`])
    > JSON is preferred, but you can also send form encoded variables by wrapping the option with `doc[]` and adding another `[]` for sub options.

- Calling `fetch()` with `URLSearchParams` ⇒ `200 OK`
  - Request `POST` with `user_credentials` → `200 OK`

```
Request URL: https://api.docraptor.com/docs?user_credentials=m7Dhrn_AsezV94C3VL-B&doc%5Btype%5D
Request Method: POST
Status Code: 200 OK
Remote Address: 54.88.97.245:443
Referrer Policy: strict-origin-when-cross-origin
origin: https://myip
priority: u=1, i
referer: https://myip/
sec-ch-ua: "Chromium";v="128", "Not;A=Brand";v="24", "Google Chrome";v="128"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
sec-fetch-dest: empty
sec-fetch-mode: cors
sec-fetch-site: cross-site
```

- Response with `Access-Control-Allow-Origin` header

```
access-control-allow-methods: GET,PUT,POST
access-control-allow-origin: https://myip
access-control-max-age: 900
cache-control: max-age=0, private, must-revalidate
content-disposition: attachment; filename="DocRaptor TestDocs.pdf"; filename*=UTF-8''DocRaptor%
content-length: 307561
content-transfer-encoding: binary
content-type: application/pdf
date: Wed, 30 Oct 2024 11:49:06 GMT
etag: W/"4b660d33d3558fb04e888493a29f3fe2"
expect-ct: max-age=86400, enforce, report-uri="https://o8095.ingest.sentry.io/api/15415/securit
referrer-policy: strict-origin-when-cross-origin
server: nginx
set-cookie: eb_tracking_id=6d84dcd6-e2f3-4657-ace7-6302d6de34f9; domain=.docraptor.com; path=/;
strict-transport-security: max-age=63072000; includeSubDomains
vary: Accept
x-content-type-options: nosniff
x-docraptor-num-pages: 31
x-download-options: noopen
x-frame-options: SAMEORIGIN
x-permitted-cross-domain-policies: none
x-request-id: fdad5347-c36b-4edb-8b29-27e7febceea3
x-runtime: 2.518433
x-xss-protection: 0
```

- Calling `curl` (and PostMan) with `authorization: Basic` & `content-type:application/json` ⇒ `200 OK`
  - Command

```
curl -v https://m7Dhrn_AsezV94C3VL-B@api.docraptor.com/docs \
  --fail --silent --show-error \
  --header "Content-Type:application/json" \
  --data '{"test": true,
           "document_url": "https://docraptor.com/examples/invoice.html",
           "type": "pdf" }' > docraptor.pdf
```

  - Request `POST` → `200 OK`

```
* Server auth using Basic with user 'm7Dhrn_AsezV94C3VL-B'
* Using Stream ID: 1 (easy handle 0x55bbbd95aa60)
* TLSv1.2 (OUT), TLS header, Supplemental data (23):
} [5 bytes data]
> POST /docs HTTP/2
> Host: api.docraptor.com
> authorization: Basic bTdEaHJuX0FzZXpWOTRDM1ZMLUI6
> user-agent: curl/7.81.0
> accept: */*
> content-type:application/json
> content-length: 115
```

  - Response with `Access-Control-Allow-Origin` header

```
< HTTP/2 200
< date: Wed, 30 Oct 2024 11:37:43 GMT
< content-type: application/pdf
< content-length: 73613
< server: nginx
< x-frame-options: SAMEORIGIN
< x-xss-protection: 0
< x-content-type-options: nosniff
< x-download-options: noopen
< x-permitted-cross-domain-policies: none
< referrer-policy: strict-origin-when-cross-origin
< expect-ct: max-age=86400, enforce, report-uri="https://o8095.ingest.sentry.io/api/15415/secur
< x-docraptor-num-pages: 1
< content-disposition: attachment; filename="doc-482759486.pdf"; filename*=UTF-8''doc-482759486
< content-transfer-encoding: binary
< vary: Accept
< etag: W/"8f32cebf1d07925d239958081b738618"
< cache-control: max-age=0, private, must-revalidate
< set-cookie: _dr_session=202db46b696109479465316a86f031b3; path=/; expires=Wed, 30 Oct 2024 16
< set-cookie: eb_tracking_id=6d84dcd6-e2f3-4657-ace7-6302d6de34f9; domain=.docraptor.com; path=
< x-request-id: f3460bdb-aced-4cf6-9434-2e7b0ff59811
< x-runtime: 1.364432
< strict-transport-security: max-age=63072000; includeSubDomains
```

- Calling `fetch()` with `form.submit()` ⇒ `200 OK`
  - Request `POST` → `200 OK`

```
Request URL: https://api.docraptor.com/docs
Request Method: POST
Status Code: 200 OK
Remote Address: 34.226.73.93:443
Referrer Policy: strict-origin-when-cross-origin
accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,
accept-encoding: gzip, deflate, br, zstd
accept-language: en-US,en;q=0.9,vi-VN;q=0.8,vi;q=0.7,ja-JP;q=0.6,ja;q=0.5
cache-control: max-age=0
content-length: 208
content-type: application/x-www-form-urlencoded
origin: https://myip
priority: u=0, i
referer: https://myip/
sec-ch-ua: "Chromium";v="128", "Not;A=Brand";v="24", "Google Chrome";v="128"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
sec-fetch-dest: document
sec-fetch-mode: navigate
sec-fetch-site: cross-site
sec-fetch-user: ?1
upgrade-insecure-requests: 1
user-agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0
```

- Response with `Access-Control-Allow-Origin` header

```
access-control-allow-methods: GET,PUT,POST
access-control-allow-origin: https://myip
access-control-max-age: 900
cache-control: max-age=0, private, must-revalidate
content-disposition: attachment; filename="DocRaptor TestDocs.pdf"; filename*=UTF-8''DocRaptor%
content-length: 307568
content-transfer-encoding: binary
content-type: application/pdf
date: Wed, 30 Oct 2024 12:51:16 GMT
etag: W/"53cb350849343d8085386f6bda50ce6a"
expect-ct: max-age=86400, enforce, report-uri="https://o8095.ingest.sentry.io/api/15415/securit
referrer-policy: strict-origin-when-cross-origin
server: nginx
set-cookie: eb_tracking_id=6d84dcd6-e2f3-4657-ace7-6302d6de34f9; domain=.docraptor.com; path=/;
strict-transport-security: max-age=63072000; includeSubDomains
x-content-type-options: nosniff
x-docraptor-num-pages: 31
x-download-options: noopen
x-frame-options: SAMEORIGIN
x-permitted-cross-domain-policies: none
x-request-id: 864d5382-98eb-4df7-89dc-7425c6929f86
x-runtime: 2.499588
x-xss-protection: 0
```

- Testing functions `toPdf_*()`

```
// DocRaptor
const DocRaptorApiKey = "m7Dhrn_AsezV94C3VL-B";
const DocRaptorUrl = `https://api.docraptor.com/docs`;
const DocRaptorRequest = {
    // Test documents are free, but watermarked **nicely** at the top & bottom of each page
    "test": true,
    // Give a name for the docs
    "name": "DocRaptor TestDocs",
    // You can supply content directly
    "document_content": "",
    // or via a URL
    //"document_url": "http://www.evopdf.com/DemoAppFiles/HTML_Files/Structured_HTML.html",
    //"javascript": true, // for HTML display before convertion
    "type": "pdf", // Output type can be "pdf" or "xls" or "xlsx"
    //"prince_options": {
        //"media": "screen" // use screen styles instead of print styles
    //}
}

const Request = {
    method: 'POST',
    headers: {
        //'Content-Type': 'application/json',
        'Content-Type': 'application/x-www-form-urlencoded',
        'Accept': '*/*',
        //'Credentials': 'include',
        //'Access-Control-Allow-Origin': '*', // for preflight OPTIONS request
```

```
        //mode: "cors", // have browser to send preflight CRLFs request
        body: '',
    }

    const makeFormElement = function(name, value) {
        var element = document.createElement("textarea")
        element.name = name
        element.value = value
        return element
    }
    async function toPdf_form() { // use form.submit(), copied from https://docraptor.com/docrapto
        let form = document.createElement("form")
        form.action = "https://api.docraptor.com/docs"
        form.method = "post"
        form.style.display = "none"

        form.appendChild(makeFormElement("user_credentials", DocRaptorApiKey))
        form.appendChild(makeFormElement("doc[type]", 'pdf'));
        form.appendChild(makeFormElement("doc[test]", true));
        form.appendChild(makeFormElement("doc[name]", 'DocRaptor TestDocs'));
        form.appendChild(makeFormElement("doc[document_url]", 'http://www.evopdf.com/DemoAppFiles/

        document.body.appendChild(form);
        form.submit()
    }

    async function toPdf_params() { // use URLSearchParams
        let url = new URL(DocRaptorUrl);
        let params = new URLSearchParams(); // url.searchParams;
        params.append("user_credentials", DocRaptorApiKey);
        params.append("doc[type]", 'pdf');
        params.append("doc[test]", true);
        params.append("doc[name]", 'DocRaptor TestDocs');
        //params.append("doc[document_url]", 'http://www.evopdf.com/DemoAppFiles/HTML_Files/Struct
        params.append("doc[document_content]", mdhtml.innerHTML + markdown_style.outerHTML + pdf_s
        let req = structuredClone(Request);
            req.body = params;

        res = await fetch(url.href, req); // , mode: "no-cors"
        console.debug('toPdf() fetch',req,res);
        if (!res.ok) {
            //console.log('Fetch error:',res);
            loadError(res.statusText);
            return;
        } else {
            message.innerHTML = '';
            message.style.display = 'none';
        }
        blob = await res.blob();
        console.debug('toPdf() fetch',params,blob);
        // blob URL will be stored in `exportUrl` and updated in cascade
        exportUrl.href = URL.createObjectURL(blob);
    }

    async function toPdf_JSON() { // use JSON
        //DocRaptorRequest.document_content = mdhtml.innerHTML + markdown_style.outerHTML + pdf_st
        DocRaptorRequest.document_url = "https://docraptor.com/examples/invoice.html";

        let req = structuredClone(Request);
            req.headers.Authorization = 'Basic '+btoa(DocRaptorApiKey+':');
            req.body = DocRaptorRequest;
        let url = DocRaptorUrl; // `${DocRaptorUrl}?user_credentials=${DocRaptorApiKey}`;
        let res = await fetch(url, req);
        console.debug('toPdf() fetch',req,res);
        if (!res.ok) {
            //console.log('Fetch error:',res);
            loadError(res.statusText);
            return;
        } else {
            message.innerHTML = '';
            message.style.display = 'none';
        }
        let blob = await res.blob();
        console.debug('toPdf() fetch',DocRaptorRequest,blob);
        // blob URL will be stored in `exportUrl` and updated in cascade
        exportUrl.href = URL.createObjectURL(blob);
    }
```

- TODO capture page into image (PNG/SVG)

- **Or, process content with vim commands**
  - remove `:logbook:` and properties

    ```
    %s/^\s*:\(logbook\|LOGBOOK\):\_.\{-}\s*:END:\n//
    %s/^\s*\w\+:: .*\n//
    ```

  - remove first tab with `Ctrl` `v`, or with command

    ```
    %s/^\t\(.*\)/\1/
    ```

  - unitemize headings

    ```
    %s/^- #\(.*\)/\r#\1\r/
    ```

  - process code block for strict conventions like in GitLab

    ```
    %s/^\(\t*\)- ```/\r\1```/|%s/  ```/```\r/|%s/\t  /\t/
    ```

  - replace items bullets with numbers: `V` select all items, then

    ```
    '<,'>s/^- \(.*\)/1. \1/
    ```

  - replace links to `assets` & `publish`

    ```
    %s#../assets/projects/java17/aal_gw/##g
    %s#(publish/projects/java17/aal_gw/\([^)]*\))#(\1.md)#g
    ```

  - Replace straight quotes with curly quotes

    ```
    %s/"\(\S\)/“\1/g|%s/\(\S\)"/\1”/g
    ```