Google Docs Cleanup Toolkit

Get Source Code

https://github.com/lsvekis/Doc-Cleanup-Toolkit

User & Developer Guide

This guide explains **what the Google Docs Cleanup Toolkit does**, **how to use it safely**, and **how each part of the system fits together**.

---

1. What This Toolkit Is

The **Google Docs Cleanup Toolkit** is a Google Apps Script project that helps you:

- Clean up messy documents
- Normalize spacing, punctuation, and formatting
- Remove AI-generated boilerplate
- Fix list, heading, and table issues
- Prepare documents for publishing, teaching, or sharing

It is designed to be:

- ✅ **Safe by default**
- 🔧 **Highly configurable**
- 🧪 **Testable with seeded content**
- 🧠 **AI-draft aware**

---

2. Who This Is For

This toolkit is ideal for:

- Educators preparing course materials
- Writers editing AI-assisted drafts
- Developers working with Google Docs automation
- Teams cleaning collaborative documents
- Anyone pasting content from multiple sources

---

3. How the Toolkit Is Structured

Repository Structure

```
doc-cleanup-toolkit/
├─ README.md                → Project overview
├─ appsscript.json          → Apps Script manifest & scopes
├─ Sidebar.html             → Optional UI panel
└─ src/
   ├─ Code.gs               → Core cleanup logic
   ├─ UI.gs                 → Menus & sidebar hooks
   └─ Tests.gs              → Test content & validation helpers
```

---

4. Installation & Setup

Option A: Manual (Recommended for most users)

1. Open a Google Doc
2. Go to **Extensions → Apps Script**
3. Copy the files from `/src` into the editor
4. Add `Sidebar.html`
5. Save and reload the document

You will now see a **“Doc Cleanup”** menu.

Option B: GitHub + clasp (Advanced)

1. Install `clasp`
2. Clone the GitHub repo
3. Create an Apps Script project
4. Push files using `clasp push`

---

5. Using the Toolkit

Main Menu

After installation, the document menu includes:

Doc Cleanup →

- **Run All (Safe Defaults)**
- **Run FULL TEST (Seeds + Runs All + Validates)**
- **Open Cleanup Panel** (optional sidebar)

---

6. Safe vs Risky Actions (Important)

Safe Actions (Default)

These **do not change document structure** and are safe to run repeatedly:

- Remove extra blank lines

- Normalize spacing and punctuation
- Remove AI boilerplate
- Remove placeholders (Lorem ipsum, TBD)
- Clean duplicate headings
- Fix smart quotes and dashes
- Normalize lists without converting them

These are included in **Run All (Safe Defaults)**.

Risky / Structure-Changing Actions

These **modify document structure** and should be run intentionally:

- Convert list items → plain text (`- item`)
- Convert numbered lists → `1. item`
- Remove emojis
- Remove entire paragraphs

These are:

- clearly labeled as **risky**
- run **last** in the full test
- excluded from safe defaults

---

7. Key Cleanup Concepts

7.1 AI Artifact Cleanup

`removeAIArtifacts_` removes phrases commonly inserted by AI tools:

Examples:

- “As an AI language model…”
- “I can’t browse the web…”
- “Here’s a draft:”

It:

- removes only the phrase
- keeps the paragraph intact
- never deletes the last paragraph

7.2 Paragraph Safety Rule

Google Docs **requires at least one paragraph per section**.

This toolkit:

- never removes the last paragraph
- replaces unremovable paragraphs with a single space
- uses safe wrappers around `removeChild`

This is why the toolkit remains stable even in complex documents.

---

8. Testing & Validation

Populate a Document with Messy Content

To generate test content:

```
populateDocWithMessyTestContent()
```

This inserts:

- spacing errors
- AI boilerplate
- duplicate headings
- fake bullets
- nested lists
- empty tables
- punctuation-only lines

No content is deleted — it only adds data.

Run the Full Test Pipeline

```
runFullCleanupTestOnActiveDoc()
```

This will:

1. Populate the document
2. Run all cleanup functions
3. Validate results
4. Log pass/fail results
5. Show a summary alert

---

9. Sidebar UI (Optional)

The sidebar provides:

- One-click **Run Safe Cleanup**
- One-click **Populate Test Content**

This is ideal for:

- non-technical users
- demos
- workshops
- classroom use

---

10. Extending the Toolkit

You can easily add:

- New cleanup rules
- Language-specific cleanup
- Preview/dry-run mode
- Highlight-instead-of-delete behavior
- Comments instead of removal
- Document summaries after cleanup

Each cleanup task is isolated and registered in a central action list.

---

11. Best Practices

- Run **Safe Defaults** first
- Review the document
- Run **risky actions** only if needed
- Use test content to validate changes
- Avoid closing the document mid-pipeline

---

12. Known Limitations

- Google Docs section rules prevent removing certain paragraphs
- Some formatting is owned by Docs and cannot be altered
- Emoji detection depends on Unicode support

All limitations are handled gracefully.

---

13. License

MIT License
Use freely in personal, educational, or commercial projects.

---

14. Where to Go Next

Suggested enhancements:

- AI-assisted cleanup suggestions
- Rule presets (Blog / Academic / Course)
- Diff preview before applying changes

- Metrics (before vs after stats)

Function Explanation

UI and orchestration

`onOpen()`

What it does: Adds a custom **Doc Cleanup** menu to Google Docs when the document is opened.

Example: After refreshing the Doc, you'll see:

- Doc Cleanup → Open Cleanup Panel
- Run All (Safe Defaults)
- Run FULL TEST

`showCleanupSidebar()`

What it does: Opens `Sidebar.html` as a Google Docs sidebar.

Example use: Click **Doc Cleanup → Open Cleanup Panel**.

`getActions()`

What it does: Returns the list of actions (id/label/category/risk) so the sidebar can populate dropdowns/buttons.

Example output (conceptual):

```
[
```

```
  { "id":"removeExtraBlankLines", "label":"Remove extra blank lines",
"category":"Structure", "risk":"safe" }
]
```

---

runAction(actionId)

What it does: Runs one cleanup action by id (from `ACTIONS_BY_ID_`).

Example: If the sidebar calls:

```
runAction('removeAIArtifacts')
```

…it runs `removeAIArtifacts_(doc)` and returns a status message.

---

runAllCleanup()

What it does: Runs your “safe defaults” pipeline (`runAllCleanupOnDoc_`).

Example: One click to clean the doc without structure-changing steps like list conversion.

---

runAllCleanupOnDoc_(doc)

What it does: The **safe pipeline**. Runs “safe” cleanups in a deliberate order.

Example effect: Fixes spacing, deletes placeholder paragraphs, normalizes punctuation, cleans headings, removes empty tables—without converting lists to plain text.

Registry

ACTIONS_LIST_

What it does: Defines every action you can run from the UI (id, label, category, risk, function pointer).

Example: This entry:

```
{ id:'removeZeroWidthChars', fn:(d)=>{ removeZeroWidthChars_(d); return '✅ ...'; } }
```

means the sidebar can run it by id.

`ACTIONS_BY_ID_`

What it does: A lookup map from action id → action object.

Example:

```
ACTIONS_BY_ID_['normalizeCommaSpacing']
```

returns the action config so `runAction` can call it.

Core helpers

`forEachParagraph_(doc, cb)`

What it does: Iterates all top-level paragraphs in the document body and calls your callback.

Example: Used by many normalizers so you don't repeat scanning logic.

`normalizeParagraphText_(doc, transformFn)`

What it does: For each paragraph, gets its text, applies `transformFn(text)`, and writes it back if changed.

Example (conceptual):

- Input: `"Hello ,world"`
- transformFn fixes comma spacing
- Output: `"Hello, world"`

`isHeading_(p)`

What it does: Returns `true` if paragraph heading is not NORMAL.

Example:

- H2 → `true`
- Normal paragraph → `false`

`isEmptyParagraph_(p)`

What it does: True if paragraph text is empty after trimming.

Example:

- `"  "` → empty
- `"Hi"` → not empty

`safeRemoveChild_(body, child)`

What it does: Attempts to remove a body child safely.

- If it can't remove (often due to section rules), it blanks paragraphs with `' '`.

Example: If a paragraph is the last in a section:

- instead of crashing, it becomes `" "`.

`clearBodyForTestingV3_(body)` (the robust one)

What it does: Clears content **for testing** without breaking section rules.

- Tries to remove everything
- If removal fails, blanks paragraphs

Example: A doc with multiple sections is reset to "mostly empty" safely (not always truly empty, but safe).

Cleanup tasks with before/after examples

1) `removeExtraBlankLines_(doc)`

What it does: Collapses multiple consecutive blank paragraphs into a single blank line.

Before:

```
Line 1

(blank)
(blank)

Line 2
```

After:

```
Line 1

Line 2
```

---

2) `removeEmptyListItems_(doc)`

What it does: Deletes list items that have no visible text.

Before:

- Item A
- (empty bullet)
- Item B

After:

- Item A
- Item B

---

3) `removeEmptyHeadings_(doc)`

What it does: Removes headings (H1–H6) that are empty/whitespace.

Before:

- H2: `"  "`

After:

- That heading is removed (or blanked safely if Docs blocks removal).

---

4) `removeConsecutivePageBreaks_(doc)`

What it does:

- Removes trailing page breaks at the end
- Removes consecutive page breaks (and page breaks at doc start)

Before:
[PageBreak]
[PageBreak]
Content…

After:
[PageBreak]
Content…

---

5) `removeExtraHorizontalRules_(doc)`

What it does: Removes consecutive horizontal rules (HR), ignoring blank lines between HRs.

Before:
HR
(blank)
HR

After:
HR

---

6) `removePunctuationOnlyParagraphs_(doc)`

What it does: Removes paragraphs that contain only punctuation/symbols.

Before:

```
!!!
```

After: Removed (or blanked to `' '` if removal isn't allowed)

---

7) `removePlaceholderParagraphs_(doc)`

What it does: Removes placeholder paragraphs like "Lorem ipsum", "TBD", "coming soon".

Before:

```
Lorem ipsum dolor sit amet
```

After: Removed/blanked.

---

8) `removeAIArtifacts_(doc)`

What it does: Removes common AI boilerplate phrases from paragraphs.

Before:

```
As an AI language model, I can't browse the web.
Here's a draft: This section explains X.
```

After:

```
This section explains X.
```

(First line might become blank → replaced with `' '`.)

---

9) `removeDuplicateConsecutiveParagraphs_(doc)`

What it does: Removes paragraphs that are repeated back-to-back.

Before:

```
DUPLICATE PARA
```

```
DUPLICATE PARA
```

After:

```
DUPLICATE PARA
```

---

10) `fixFakeBulletsToDashes_(doc)`

What it does: Converts “fake bullet” characters at the start of paragraphs into `- `.

Before:

```
• item one
✓ done
– item two
```

After:

```
- item one
- done
- item two
```

---

11) `flattenListNesting_(doc)`

What it does: Sets every list item nesting level to 0 (no indentation).

Before:

- Item A
    - Nested item B (level 2)

After:

- Item A

- Nested item B (level 0)

---

12) `removeListItemsWithOnlyPunctuation_(doc)`

What it does: Removes list items whose text is only punctuation/symbols.

Before:

- ...
- Item A

After:

- Item A

---

13) `removeZeroWidthChars_(doc)`

What it does: Removes invisible characters: `\u200B-\u200D` and `\uFEFF`.

Before: (looks normal but causes weird cursor/spacing)

`Hello‍World`

After:

`HelloWorld`

---

14) `normalizeNbsp_(doc)`

What it does: Converts NBSP (`\u00A0`) into a normal space.

Before:
```
Hello\u00A0World
```

After:
```
Hello World
```

---

15) `convertTabsToSpaces_(doc)`

What it does: Replaces tabs (`\t`) with spaces.

Before:
```
A\t\tB\tC
```

After:
```
A B C
```

---

16) `trimLeadingTrailingWhitespace_(doc)`

What it does: Removes leading and trailing spaces/tabs on each paragraph.

Before:
```
" Hello world "
```

After:
```
"Hello world"
```

---

17) `collapseMultipleSpaces_(doc)`

What it does: Collapses 2+ spaces into a single space.

Before:
```
"Hello world"
```

After:
```
"Hello world"
```

---

18) `removeSpacesBeforePunctuation_(doc)`

What it does: Fixes `"word , word"` → `"word, word"`.

Before:
`"Hi , there ."`

After:
`"Hi, there."`

---

19) `ensureSpaceAfterPunctuation_(doc)`

What it does: Ensures punctuation is followed by a space when a letter follows.

Before:
`"Hello,world.This;works:now"`

After:
`"Hello, world. This; works: now"`

---

20) `normalizeEllipses_(doc)`

What it does: Converts `....` or longer to `...`.

Before:
`"Wait.... what......"`

After:
`"Wait... what..."`

---

21) `fixRepeatedPunctuationRuns_(doc)`

What it does: Collapses runs like `!!!` to `!` (same for `??`, `,,`, `..` etc.)

Before:
```
"No!!! Really???"
```

After:
```
"No! Really?"
```

---

22) `normalizeSmartPunctuation_(doc)`

What it does: Converts smart quotes/dashes to plain ASCII.

Before:
```
“Hello” — ‘world’
```

After:
```
"Hello" - 'world'
```

---

23) `normalizeDashSpacing_(doc)`

What it does: Standardizes spacing around `-`.

Before:
```
"A-B","A -B","A- B"
```

After:
```
"A - B"
```

---

24) `normalizeQuoteSpacing_(doc)`

What it does: Fixes spacing around quotes and punctuation near quotes.

Before:
```
Quotes spacing:"Hello" , "World" .
```

After:
```
Quotes spacing: "Hello", "World".
```

---

25) `removeDoubleCommasPeriods_(doc)`

What it does: Converts `,,` → `,` and `..` → `.` (but preserves `...`).

Before:
`"Hello,, world.."`

After:
`"Hello, world."`

---

26) `normalizeApostrophes_(doc)`

What it does: Converts curly apostrophe `’` to `'`.

Before: `Don’t`
After: `Don't`

---

27) `normalizeSlashesSpacing_(doc)`

What it does: Ensures spacing around `/`.

Before:
`A/B C /D E/ F`

After:
`A / B C / D E / F`

---

28) `normalizeColonSemicolonSpacing_(doc)`

What it does: Ensures space after `:` and `;` when followed by a letter.

Before:
`"Note:This;works"`

After:
`"Note: This; works"`

---

29) `removeSpaceAfterOpeningBracket_(doc)`

What it does: Removes space after `( [ {`.

Before: `( hello`
After: `(hello`

---

30) `removeSpaceBeforeClosingBracket_(doc)`

What it does: Removes space before `) ] }`.

Before: `hello )`
After: `hello)`

---

31) `normalizeUnicodeBulletsInText_(doc)`

What it does: Converts `•` or `·` inside normal text to `-`.

Before:
`Inline bullets: • one · two`

After:
`Inline bullets: - one - two`

---

32) `reduceHeadingSizes_(doc)`

What it does: Reduces headings by one level:
H1→H2, H2→H3, … H5→H6.

Before: H1 “Title”
After: H2 “Title”

---

33) `removeDuplicateConsecutiveHeadings_(doc)`

What it does: Removes back-to-back headings with identical text.

Before:
H2 “Intro”
H2 “Intro”

After:
H2 “Intro”

---

34) `normalizeHeadingSpacing_(doc)`

What it does:

- Collapses multiple blank lines *before* headings to max 1
- Ensures a blank line *after* headings

Before:
(blank)
(blank)
H2 Title
Next paragraph immediately

After:
(blank)
H2 Title
(blank)
Next paragraph…

---

35) `titleCaseAllCapsHeadings_(doc)`

What it does: If a heading is ALL CAPS, converts it to Title Case.

Before: `THIS IS A HEADING`
After: `This Is A Heading`

---

36) `removeShortHeadings_(doc, maxLen)`

What it does: Removes headings whose text length is `<= maxLen`.

Before: H4 “A”
After: removed/blanked

---

37) `removeEmptyTables_(doc)`

What it does: Removes tables where every cell is empty/whitespace.

Before: 2x2 table of spaces
After: table removed (or left if Docs blocks removal, depending on section rules)

---

38) `convertBulletsToDashes_(doc)` (risky)

What it does: Converts real Docs list items into plain paragraphs starting with `- `.

Before:

- Real bullet A
- Real bullet B

After:

```
- Real bullet A
- Real bullet B
```

(You lose real list structure.)

---

39) `convertNumberedListsToPlainText_(doc)` (risky)

What it does: Converts numbered list items into plain paragraphs starting with `1.`, `2.`, …

Before: a numbered list
After: text lines like `1. Step one`

---

40) `removeEmojis_(doc)` (risky)

What it does: Removes emoji characters using Unicode properties.

Before: `Great job ✅🔥`
After: `Great job`

---

41) `removeLeadingHyphenOnlyLines_(doc)`

What it does: Removes paragraphs that are only hyphens.

Before: `---`
After: removed/blanked

---

42) `normalizeParenthesesSpacing_(doc)`

What it does: Fixes spaces just inside parentheses.

Before: `( hello )`
After: `(hello)`

---

43) `normalizeCommaSpacing_(doc)`

What it does: Fixes spacing around commas.

Before: `hello ,world,again`
After: `hello, world, again`

---

44) `removeDuplicateConsecutiveListItems_(doc)`

What it does: Removes consecutive identical list items.

Before:

- Duplicate LI
- Duplicate LI

After:

- Duplicate LI

---

45) `removeParagraphsWithOnlyNumbers_(doc)`

What it does: Removes paragraphs like `123`.

Before: `123`
After: removed/blanked

Testing and seeding

`populateDocWithMessyTestContent()`

What it does: Appends messy content to trigger cleanup functions (safe; no deletions).

Example: Adds fake bullets, extra spaces, duplicate headings, empty list items, empty tables, etc.

---

seedFullTestContentInActiveDoc_()

What it does: Your “full seed” builder used by your full test run (depends on how you clear/reset first).

---

runAllFunctionsOnDoc_(doc)

What it does: Runs safe pipeline, then extra safe tasks, then risky tasks last.

---

validateFullTest_(doc)

What it does: Reads full body text and checks expected outcomes (placeholder removed, tabs removed, no list items remain after conversion, etc.)

Example output lines:

- ✅ Removed Lorem ipsum
- ✅ Converted tabs
- ❌ Empty table removed (heuristic)