/pptx

Source:
`~/.claude/skills/pptx/SKILL.md`

---

name: pptx description: "Use this skill any time a .pptx file is involved in any way — as input, output, or both. This includes: creating slide decks, pitch decks, or presentations; reading, parsing, or extracting text from any .pptx file (even if the extracted content will be used elsewhere, like in an email or summary); editing, modifying, or updating existing presentations; combining or splitting slide files; working with templates, layouts, speaker notes, or comments. Trigger whenever the user mentions "deck,"

"slides," "presentation," or references a .pptx filename, regardless of what they plan to do with the content afterward. If a .pptx file needs to be opened, created, or touched, use this skill." license: Proprietary. LICENSE.txt has complete terms

PPTX Skill

Quick Reference

Task	Guide
Read/analyze content	`python -m markitdown presentation.pptx`
Edit or create from template	Read editing.md
Create from scratch	Read pptxgenjs.md

Reading Content

```
# Text extraction
python -m markitdown presentation.pptx

# Visual overview
python scripts/thumbnail.py presentation.pptx

# Raw XML
python scripts/office/unpack.py presentation.pptx unpacked/
```

Editing Workflow

Read editing.md for full details.

1. Analyze template with `thumbnail.py`
2. Unpack → manipulate slides → edit content → clean → pack

Creating from Scratch

Read pptxgenjs.md for full details.

Use when no template or reference presentation is available.

Design Ideas

Don't create boring slides. Plain bullets on a white background won't impress anyone. Consider ideas from this list for each slide.

Before Starting

- **Pick a bold, content-informed color palette**: The palette should feel designed for THIS topic. If swapping your colors into a completely different presentation would still "work," you haven't made specific enough choices.
- **Dominance over equality**: One color should dominate (60-70% visual weight), with 1-2 supporting tones and one sharp accent. Never give all colors equal weight.
- **Dark/light contrast**: Dark backgrounds for title + conclusion slides, light for content ("sandwich" structure). Or commit to dark throughout for a premium feel.
- **Commit to a visual motif**: Pick ONE distinctive element and repeat it — rounded image frames, icons in colored circles, thick single-side borders. Carry it across every slide.

Color Palettes

Choose colors that match your topic — don't default to generic blue. Use these palettes as inspiration:

Theme	Primary	Secondary	Accent
Midnight Executive	`1E2761` (navy)	`CADCFC` (ice blue)	`FFFFFF` (white)
Forest & Moss	`2C5F2D` (forest)	`97BC62` (moss)	`F5F5F5` (cream)
Coral Energy	`F96167` (coral)	`F9E795` (gold)	`2F3C7E` (navy)
Warm Terracotta	`B85042` (terracotta)	`E7E8D1` (sand)	`A7BEAE` (sage)
Ocean Gradient	`065A82` (deep blue)	`1C7293` (teal)	`21295C` (midnight)
Charcoal Minimal	`36454F` (charcoal)	`F2F2F2` (off-white)	`212121` (black)
Teal Trust	`028090` (teal)	`00A896` (seafoam)	`02C39A` (mint)
Berry & Cream	`6D2E46` (berry)	`A26769` (dusty rose)	`ECE2D0` (cream)
Sage Calm	`84B59F` (sage)	`69A297` (eucalyptus)	`50808E` (slate)
Cherry Bold	`990011` (cherry)	`FCF6F5` (off-white)	`2F3C7E` (navy)

For Each Slide

Every slide needs a visual element — image, chart, icon, or shape. Text-only slides are forgettable.

Layout options:

- Two-column (text left, illustration on right)
- Icon + text rows (icon in colored circle, bold header, description below)
- 2x2 or 2x3 grid (image on one side, grid of content blocks on other)
- Half-bleed image (full left or right side) with content overlay

Data display:

- Large stat callouts (big numbers 60-72pt with small labels below)
- Comparison columns (before/after, pros/cons, side-by-side options)
- Timeline or process flow (numbered steps, arrows)

Visual polish:

- Icons in small colored circles next to section headers
- Italic accent text for key stats or taglines

Typography

Choose an interesting font pairing — don't default to Arial. Pick a header font with personality and pair it with a clean body font.

Header Font	Body Font
Georgia	Calibri
Arial Black	Arial
Calibri	Calibri Light
Cambria	Calibri
Trebuchet MS	Calibri
Impact	Arial
Palatino	Garamond
Consolas	Calibri

Element	Size
Slide title	36-44pt bold
Section header	20-24pt bold
Body text	14-16pt
Captions	10-12pt muted

Spacing

- 0.5" minimum margins
- 0.3-0.5" between content blocks
- Leave breathing room—don't fill every inch

Avoid (Common Mistakes)

- **Don't repeat the same layout** — vary columns, cards, and callouts across slides
- **Don't center body text** — left-align paragraphs and lists; center only titles
- **Don't skimp on size contrast** — titles need 36pt+ to stand out from 14-16pt body
- **Don't default to blue** — pick colors that reflect the specific topic
- **Don't mix spacing randomly** — choose 0.3" or 0.5" gaps and use consistently
- **Don't style one slide and leave the rest plain** — commit fully or keep it simple throughout
- **Don't create text-only slides** — add images, icons, charts, or visual elements; avoid plain title + bullets
- **Don't forget text box padding** — when aligning lines or shapes with text edges, set `margin: 0` on the text box or offset the shape to account for padding
- **Don't use low-contrast elements** — icons AND text need strong contrast against the background; avoid light text on light backgrounds or dark text on dark backgrounds
- **NEVER use accent lines under titles** — these are a hallmark of AI-generated slides; use whitespace or background color instead

QA (Required)

Assume there are problems. Your job is to find them.

Your first render is almost never correct. Approach QA as a bug hunt, not a confirmation step. If you found zero issues on first inspection, you weren't looking hard enough.

Content QA

```
python -m markitdown output.pptx
```

Check for missing content, typos, wrong order.

When using templates, check for leftover placeholder text:

```
python -m markitdown output.pptx | grep -iE "xxxx|lorem|ipsum|this.*(page|slide).*layout"
```

If grep returns results, fix them before declaring success.

Visual QA

⚠ **USE SUBAGENTS** — even for 2-3 slides. You've been staring at the code and will see what you expect, not what's there. Subagents have fresh eyes.

Convert slides to images (see Converting to Images), then use this prompt:

```
Visually inspect these slides. Assume there are issues — find them.

Look for:
- Overlapping elements (text through shapes, lines through words, stacked elements)
- Text overflow or cut off at edges/box boundaries
- Decorative lines positioned for single-line text but title wrapped to two lines
- Source citations or footers colliding with content above
- Elements too close (< 0.3" gaps) or cards/sections nearly touching
- Uneven gaps (large empty area in one place, cramped in another)
- Insufficient margin from slide edges (< 0.5")
- Columns or similar elements not aligned consistently
- Low-contrast text (e.g., light gray text on cream-colored background)
```

```
- Low-contrast icons (e.g., dark icons on dark backgrounds without a contrasting circle)
- Text boxes too narrow causing excessive wrapping
- Leftover placeholder content

For each slide, list issues or areas of concern, even if minor.

Read and analyze these images:
1. /path/to/slide-01.jpg (Expected: [brief description])
2. /path/to/slide-02.jpg (Expected: [brief description])

Report ALL issues found, including minor ones.
```

Verification Loop

1. Generate slides → Convert to images → Inspect
2. **List issues found** (if none found, look again more critically)
3. Fix issues
4. **Re-verify affected slides** — one fix often creates another problem
5. Repeat until a full pass reveals no new issues

Do not declare success until you've completed at least one fix-and-verify cycle.

Converting to Images

Convert presentations to individual slide images for visual inspection:

```
python scripts/office/soffice.py --headless --convert-to pdf output.pptx
pdftoppm -jpeg -r 150 output.pdf slide
```

This creates `slide-01.jpg`, `slide-02.jpg`, etc.

To re-render specific slides after fixes:

```
pdftoppm -jpeg -r 150 -f N -l N output.pdf slide-fixed
```

Dependencies

- `pip install "markitdown[pptx]"` - text extraction

- `pip install Pillow` - thumbnail grids
- `npm install -g pptxgenjs` - creating from scratch
- LibreOffice (`soffice`) - PDF conversion (auto-configured for sandboxed environments via `scripts/office/soffice.py`)
- Poppler (`pdftoppm`) - PDF to images

Revision #5
Created 2026-02-18 08:39:52 UTC by John
Updated 2026-06-21 20:00:52 UTC by John