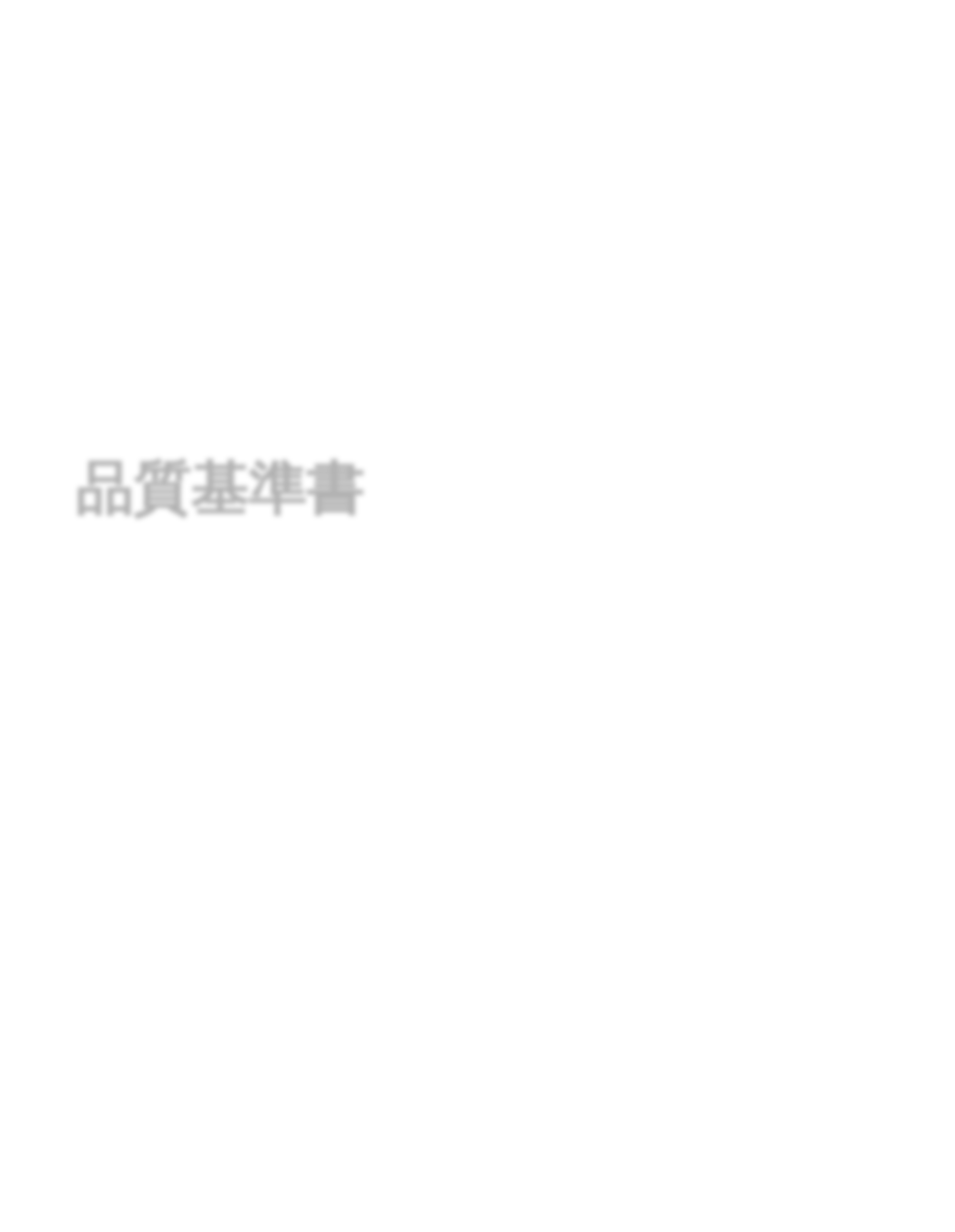
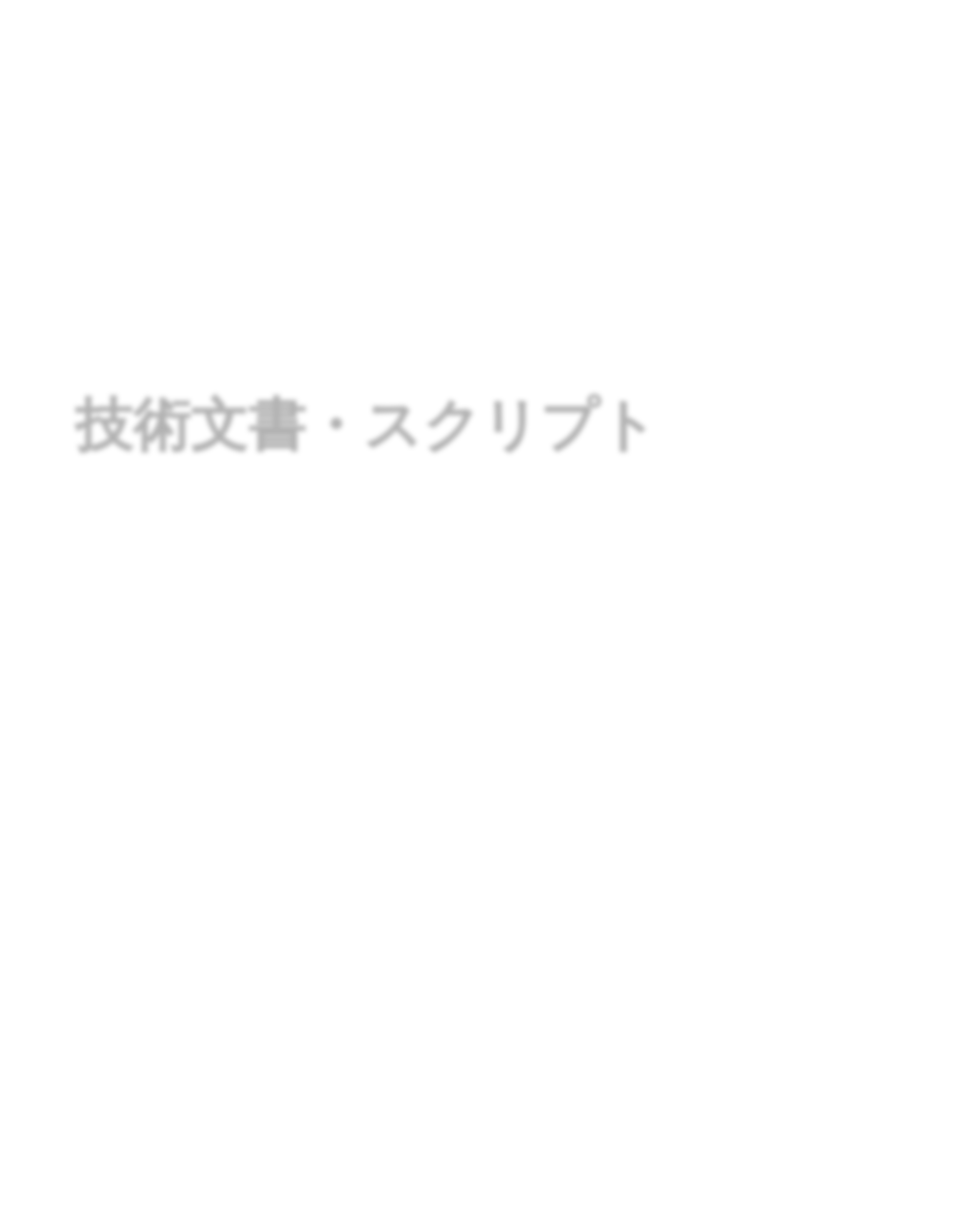
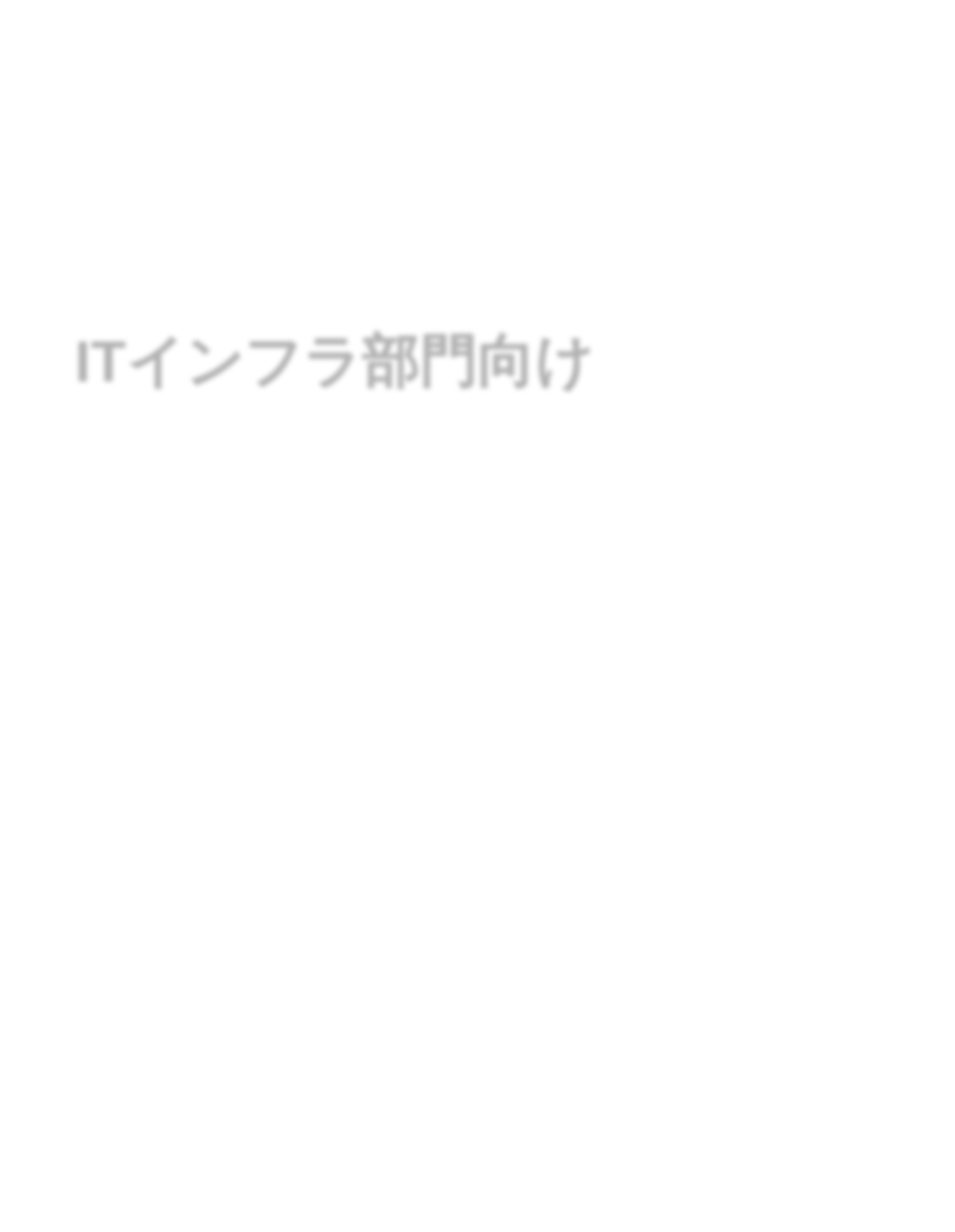
品質基準書 **v2.0**



IT INFRASTRUCTURE STANDARDS

**IT**インフラ部門向け 技術文書・スクリプト品質基準書

システム運用の信頼性向上と

開発効率最適化のための統合ガイドライン

発行日: 2025年10月2日

対象: ITインフラストラクチャ部門分類: 技術標準・品質管理

* 基本原則：スタイルの活用



目次

* 書式詳細
* レイアウトと構成要素





* 命名規則と変数
* コードスタイルと可読性
* エラーハンドリング
* ファイル・ディレクトリ構造





* 命名規則と関数設計

**IT**インフラ部門向け 技術文書・スクリプト品質基準書

本基準書は、ITインフラ部門で作成されるすべての技術文書および自動化スクリプトの品質、一貫性、保守性を確保することを目的とします。統一された基準に従うことで、チーム全体の生産性を向上させ、属人化を排除し、ナレッジの円滑な継承を促進します。

目的： 全ての成果物において、誰が作成しても、誰が読んでも理解しやすく、メンテナンスが容易な状態を維持すること。

第**1**部**: Word**文書 書式基準

本セクションでは、報告書、設計書、手順書など、Wordで作成されるすべての技術文書に適用される書式基準を定めます。文書のプロフェッショナルな外観と可読性を保証します。

基本原則：スタイルの活用

手動での書式設定（フォントサイズや色を個別に変更すること）は原則として禁止し、Wordの「スタイル」機能を全面的に活用します。これにより、文書全体で一貫したデザインを保ち、後からの書式変更を容易にします。

効率性： スタイルを変更するだけで、文書内の該当箇所すべてに一括で変更が適用されます。

一貫性： 文書全体で統一された見出しレベル、フォント、間隔を維持できます。

構造化： スタイルを利用することで、目次の自動生成やナビゲーションウィンドウでの文書構造の可視化が可能になります。

書式詳細

以下に、各要素に適用する標準スタイルを定義します。これらは推奨値であり、プロジェクトの特性や顧客の要件に応じてカスタマイズ可能です。

見出し **(Heading)**

見出しは文書の論理構造を示す最も重要な要素です。階層を正しく使い分けることで、読者は内容を素早く把握できます。[見出しは本文より目立つデザインで作成します。](https://kocossa.com/word_style/)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| スタイル名 | 用途 | フォント | サイズ | 色 | その他 |
| 見出し  **1** | 章タイトル (章番号を付与) | メイリオ or Yu Gothic UI | 16 pt | 黒 (テキスト 1) | 太字、段落前  24 pt |
| 見出し  **2** | 節タイトル (1.1) | メイリオ or Yu Gothic UI | 14 pt | 黒 (テキスト 1) | 太字、段落前  18 pt |
| 見出し  **3** | 項タイトル (1.1.1) | メイリオ or Yu Gothic UI | 12 pt | 黒 (テキスト 1) | 太字、段落前  12 pt |
| 見出し  **4-6** | 補助的な小見出し | メイリオ or Yu Gothic UI | 11 pt | 濃い灰色 (テキスト 2) | 必要に応じて使用 |

本文 **(Body Text)**

本文は文書の主要な内容を記述する部分です。読みやすさを最優先し、適切なフォントと行間を設定します。[ビジネス文書の本文は通常10〜12ポイントのフォントが使用されます。](https://edu.gcfglobal.org/en/business-communication/choosing-fonts-for-business-documents/1/)

フォント：游明朝 (Yu Mincho) or MS 明朝

サイズ：10.5 pt 〜 11 pt

色：黒 (自動)

行間：1.5行 〜 1.8行 (固定値ではなく倍数で指定)

配置：両端揃え (日本語文書の場合)

## 表・図・箇条書き

情報を整理し、視覚的に分かりやすく伝えるために、これらの要素を効果的に使用します。

表 **(Table):**

Wordの「表のデザイン」機能を使用し、グリッド線を基本としたシンプルなスタイルを適用します。

ヘッダー行は背景色を変え、太字に設定して区別します。

表のタイトルは表の上に「表 1: (タイトル)」の形式で記述します。

図 **(Figure):**

図のタイトルは図の下に「図 1: (タイトル)」の形式で記述します。解像度の低い画像は使用を避け、鮮明なものを挿入します。

箇条書き **(List):**

順序が重要な場合は番号付きリスト (1, 2, 3...) を使用します。

順序が重要でない場合は黒丸 (●) などの記号付きリストを使用します。各項目の文体（〜ですます調、体言止めなど）を統一します。

レイアウトと構成要素

### 目次 (Table of Contents)

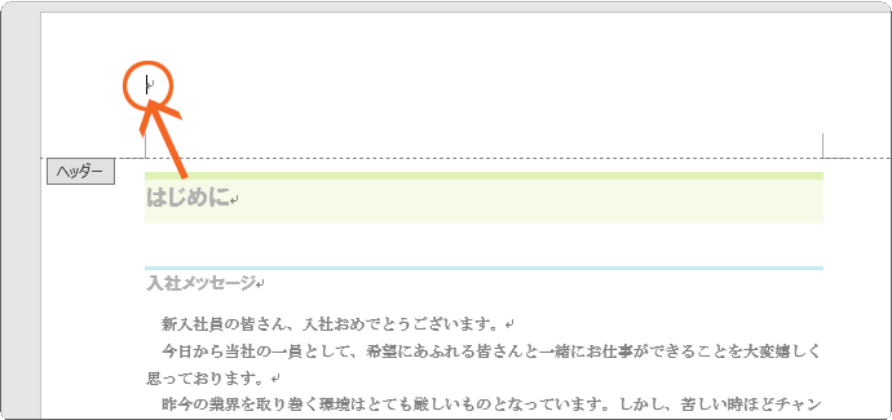
文書の冒頭（表紙の次ページ）に、Wordの目次自動生成機能を用いて作成します。見出しスタイル（見出し1〜3）を正しく適用していれば、数クリックで正確な目次が生成・更新できます。[文書を編集した後は、必ず目次を更新してください。](https://support.microsoft.com/ja-jp/office/%E7%9B%AE%E6%AC%A1%E3%82%92%E6%9B%B4%E6%96%B0%E3%81%99%E3%82%8B-6c727329-d8fd-44fe-83b7-fa7fe3d8ac7a)

ヘッダーとフッター **(Header & Footer)**

全ページで一貫した情報を提供するために、ヘッダーとフッターを効果的に利用します。

ヘッダー：文書タイトルや章タイトルを配置します。右端に会社のロゴを配置することも推奨されます。

フッター：中央または右端にページ番号を必ず挿入します。「ページ X / Y」形式が望ましいです。



*Word*のヘッダー領域にカーソルを合わせ、文書タイトルやロゴを挿入する

第**2**部**: BAT**ファイル コーディング規約

BAT（バッチ）ファイルは、Windows環境での定型作業を自動化するための強力なツールで す。しかし、記述の自由度が高い反面、無秩序に作成すると保守性が著しく低下します。本規約は、読みやすく、再利用可能で、安定したバッチファイルを作成するための指針です。

命名規則と変数

変数命名

変数の役割が直感的にわかるような命名を心がけます。バッチファイルの変数は大文字と小文字を区別しませんが、一貫した命名規則を採用することで可読性が向上します。[GitHubのスタイルガイドでは、snake\_caseまたはlowerCamelCaseが推奨されています。](https://github.com/trgwii/Batch-Style-Guide)

推奨： snake\_case (例: log\_file\_path ) または lowerCamelCase (例: logFilePath )。プロジェクト内で統一してください。

禁止： 意味のない変数名 (例: a , temp1 )、数字で始まる変数名。

スコープ： setlocal と endlocal を使用して変数のスコープを限定し、意図しない副作用を防ぎます。

@echo off setlocal

set "log\_file\_path=C:\logs\app.log"

echo Log file is at %log\_file\_path% endlocal



# コードスタイルと可読性

## コメント

コメントは「なぜ」その処理が必要なのかを説明するために記述します。 rem コマンドを公式なコメント行として使用します。:: はラベルとして機能するため、 for ループ内などで予期せぬ動作を引き起こす可能性があります。[そのため、`rem`の使用が推奨されます。](https://github.com/trgwii/Batch-Style-Guide)

rem ==================================

rem バックアップ処理を開始 rem 作成者: Taro Yamada rem 作成日: 2025-10-02

rem ==================================

rem 処理内容: source\_dir から dest\_dir へファイルをコピーする

xcopy "%source\_dir%" "%dest\_dir%" /E /H /C /I

## インデントとブロック

if 文やfor 文のブロックは括弧 () で囲み、処理内容をインデント（字下げ）します。これにより、コードの構造が視覚的に分かりやすくなります。

if exist "%target\_file%" (

echo Deleting existing file... del "%target\_file%"

) else (

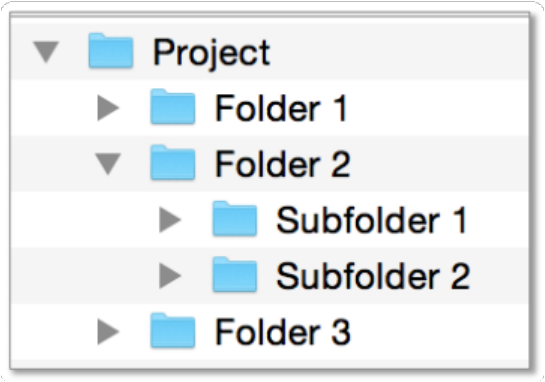
echo File not found. Skipping deletion.

)

# エラーハンドリング

コマンド実行後のエラーレベル ( %ERRORLEVEL% ) を確認し、処理が成功したか失敗したかを判定します。重要な処理の後には必ずエラーチェックを実装し、失敗した場合の処理（ログ出力、処理の中断など）を明確に記述します。

xcopy "%source\_dir%" "%dest\_dir%" /E /H /C /I



if %ERRORLEVEL% neq 0 (

echo [ERROR] File copy failed. Errorlevel: %ERRORLEVEL% exit /b 1

) else (

echo [SUCCESS] File copy completed.

)

# ファイル・ディレクトリ構造

関連するスクリプトや設定ファイル、ログファイルは、一貫したディレクトリ構造で管理します。これにより、プロジェクトの見通しが良くなり、管理が容易になります。

メインスクリプト： プロジェクトのルートディレクトリに配置します (例:

run\_backup.bat )。

ライブラリ**/**モジュール： 共通処理を記述したスクリプトは lib や scripts フォルダにまとめ、 call コマンドで呼び出します。

設定ファイル： 環境変数などを定義するファイルは conf フォルダに配置します。

ログファイル： スクリプトの実行ログは logs フォルダに出力します。

論理的なフォルダ階層の例。スクリプト、設定、ログなどを分離して管理する

第**3**部**: PowerShell**スクリプト コーディング規約

PowerShellは、Windowsの管理と自動化における標準的なシェルおよびスクリプト言語で



す。オブジェクト指向の特性を活かし、一貫性のある規約に従うことで、非常に強力で再利用性の高いスクリプトを作成できます。

命名規則と関数設計

関数命名

PowerShellの最大の特長の一つは、「動詞-名詞」形式の命名規則です。この規則に従うことで、関数の目的が明確になります。承認された動詞は Get-Verb コマンドで確認できます。

推奨： Get-ServiceStatus , Set-ServerConfig , Test-Path

非推奨： CheckPath , ServerConfig , do\_something

大文字小文字： PascalCase (例: Get-Process ) を使用します。

変数命名

変数名もPascalCaseまたはcamelCaseで記述し、その役割が明確にわかるようにします。

推奨： $targetComputerName , $logPath

非推奨： $a , $temp

# パラメータ定義

高度な関数 (Advanced Functions) を作成し、[Parameter()] 属性を使用してパラメータの動作を厳密に定義します。これにより、スクリプトの堅牢性と使いやすさが向上します。

function Get-DiskSpace { [CmdletBinding()] param (

[Parameter(Mandatory=$true, ValueFromPipeline=$true)] [string[]]$ComputerName,

[Parameter(Mandatory=$false)] [string]$DriveLetter = "C"

)

process {

# ... 処理 ...

}

}

**Mandatory:** 必須パラメータには [Parameter(Mandatory=$true)] を指定します。

**Pipeline Input:** パイプラインからの入力を受け付ける場合は ValueFromPipeline=$true を指定します。

**Switch Parameter:** オン/オフを切り替えるフラグには [switch] 型を使用します。

# エラーハンドリング

PowerShellのエラーには「Terminating (致命的)」と「Non-terminating (非致命的)」の2種類があります。 try/catch/finally ブロックは致命的なエラーのみを捕捉します。非致命的なエラーを捕捉するには、コマンド実行時に -ErrorAction Stop を指定して致命的なエラーに変換する必要があります。

try {

# Get-ChildItemは通常、非致命的エラーを発生させる

$files = Get-ChildItem -Path "C:\non\_existent\_folder" -ErrorAction Stop Write-Host "Files found: $($files.Count)"

}

catch {

# $\_ は現在の例外オブジェクトを保持する

Write-Error "An error occurred: $($\_.Exception.Message)"

}

finally {

Write-Host "Disk space check finished."

}

# ドキュメントとコメント

すべての関数には、コメントベースのヘルプを記述します。これにより、 Get-Help コマン ドで関数の使い方、パラメータ、使用例などを確認できるようになります。これは、スクリプトの再利用性と共有を促進する上で非常に重要です。

<#

.SYNOPSIS

指定されたコンピュータのディスク空き容量を取得します。

.DESCRIPTION

リモートまたはローカルコンピュータの指定されたドライブの空き容量（GB）と使用率（%）を返します。

.PARAMETER ComputerName

対象のコンピュータ名。

.PARAMETER DriveLetter

対象のドライブ文字（例: "C"）。デフォルトは "C" です。

.EXAMPLE

Get-DiskSpace -ComputerName "SERVER01", "SERVER02"

.EXAMPLE

"localhost" | Get-DiskSpace -DriveLetter "D"

#>

function Get-DiskSpace { # ... 関数の実装 ...

}

第**4**部**:** 基準書のカスタマイズと運用

本基準書は、そのまま利用するだけでなく、各組織やプロジェクトの固有の要件に合わせてカスタマイズすることを前提としています。このセクションでは、本Wordドキュメントをテンプレートとして効果的に運用する方法について解説します。

**Word**テンプレートとしての保存と配布

本ドキュメントを組織の標準として利用するために、Wordテンプレートファイル ( .dotx ) として保存します。これにより、新規文書を作成する際に、常に定義済みのスタイルや書式が適用された状態で開始できます。

1. カスタマイズ： 次のセクションに従い、ロゴの挿入や色の変更など、必要なカスタマイズを行います。
2. 名前を付けて保存： [ファイル] > [名前を付けて保存] を選択します。
3. ファイルの種類： ファイルの種類で「Word テンプレート (\*.dotx)」を選択します。
4. 保存： テンプレート名（例: IT-Standard-Template.dotx ）を入力して保存します。保存場所は自動的に「カスタム Office テンプレート」フォルダになります。
5. 配布と利用： 保存した .dotx ファイルをチームメンバーに配布します。メンバーは、このファイルをダブルクリックするか、Wordの [ファイル] > [新規] > [個人用] または [カスタム]

から選択することで、基準に準拠した新規文書を作成できます。

# デザインのカスタマイズ

## 会社ロゴの挿入

ヘッダーを編集モードで開き（上部余白をダブルクリック）、会社のロゴ画像を挿入します。画像のレイアウトオプションを「前面」または「背面」に設定し、適切な位置（通常は右上）に配置します。

## カラーテーマの変更

文書全体のカラーテーマ（見出しの色、アクセントカラーなど）は、Wordのテーマ機能で一括変更できます。

1. [デザイン] タブを開きます。
2. [色] をクリックし、プリセットの配色を選択するか、 [色のカスタマイズ] を選択して独自の配色を作成します。
3. 作成した配色は名前を付けて保存でき、他の文書でも再利用可能です。

追加規則の挿入

本基準書に記載されていない組織固有のルール（例: 特定のツールの使用方法、セキュリティポリシーなど）は、適切なセクションに追記してください。追記する際も、本基準書で定義されたスタイル（見出し、本文、箇条書きなど）を適用し、一貫性を保ちます。

改訂履歴の管理

基準書は生きたドキュメントです。技術の進化や組織の変化に伴い、定期的な見直しと更新が必要です。文書の信頼性を維持するため、変更履歴を管理することが重要です。[マニュアルを改訂する際は、どこを変更したのか改訂履歴を記録します。](https://media.tebiki.jp/161)

文書の先頭（目次の前または後）に「改訂履歴」の表を設けることを推奨します。

バージョン

改訂日

改訂内容

改訂者

1.0

2025-10-02

初版作成

（担当者名）

1.1

YYYY-MM-DD

（例: PowerShellのエラーハンドリング規約を更新）

（担当者名）

Wordの「変更履歴の記録」機能（[校閲]タブ）を活用することで、誰がいつどこを修正したかを視覚的に追跡することも可能です。

参考資料

1. バッチファイル(BATファイル)とは？作り方や書き方を初心者向けに ... <https://it-trend.jp/development_tools/article/32-0040>
2. 新しいスタイルをカスタマイズまたは作成する - Microsoft サポート [https://support.microsoft.com/ja-jp/office/%E6%96%B0%E3%81%97%E3%81%84%E3%82%B 9%E3%82%BF%E3%82%A4%E3%83%AB%E3%82%92%E3%82%AB%E3%82%B9%E3%82% BF%E3%83%9E%E3%82%A4%E3%82%BA%E3%81%BE%E3%81%9F%E3%81%AF%E4%B D%9C%E6%88%90%E3%81%99%E3%82%8B-d38d6e47-f6fc-48eb-a607-1eb120dec563](https://support.microsoft.com/ja-jp/office/%E6%96%B0%E3%81%97%E3%81%84%E3%82%B9%E3%82%BF%E3%82%A4%E3%83%AB%E3%82%92%E3%82%AB%E3%82%B9%E3%82%BF%E3%83%9E%E3%82%A4%E3%82%BA%E3%81%BE%E3%81%9F%E3%81%AF%E4%BD%9C%E6%88%90%E3%81%99%E3%82%8B-d38d6e47-f6fc-48eb-a607-1eb120dec563)
3. Lists - Microsoft Style Guide

<https://learn.microsoft.com/en-us/style-guide/scannable-content/lists>

1. about\_Functions\_Advanced\_Par...

[https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about\_functi ons\_advanced\_parameters?view=powershell-7.5](https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_functions_advanced_parameters?view=powershell-7.5)

1. Everything you wanted to know about exceptions - PowerShell

[https://learn.microsoft.com/en-us/powershell/scripting/learn/deep-dives/everything-about-exceptio ns?view=powershell-7.5](https://learn.microsoft.com/en-us/powershell/scripting/learn/deep-dives/everything-about-exceptions?view=powershell-7.5)

1. Change a theme and make it the default in Word or Excel

[https://support.microsoft.com/en-us/office/change-a-theme-and-make-it-the-default-in-word-or-exc el-c846f997-968e-4daa-b2d4-42bd2afef904](https://support.microsoft.com/en-us/office/change-a-theme-and-make-it-the-default-in-word-or-excel-c846f997-968e-4daa-b2d4-42bd2afef904)

1. Business Communication: Choosing Fonts for Business Documents <https://edu.gcfglobal.org/en/business-communication/choosing-fonts-for-business-documents/1/>
2. 【マニュアル改訂方法】表紙の書き方から履歴の残し方まで！目的 ... <https://media.tebiki.jp/161>
3. Wordの見出しデザインを一括設定する方法 - kocossa blog

<https://kocossa.com/word_style/>

1. trgwii/Batch-Style-Guide - GitHub <https://github.com/trgwii/Batch-Style-Guide>
2. 目次を更新する - Microsoft サポート

[https://support.microsoft.com/ja-jp/office/%E7%9B%AE%E6%AC%A1%E3%82%92%E6%9B% B4%E6%96%B0%E3%81%99%E3%82%8B-6c727329-d8fd-44fe-83b7-fa7fe3d8ac7a](https://support.microsoft.com/ja-jp/office/%E7%9B%AE%E6%AC%A1%E3%82%92%E6%9B%B4%E6%96%B0%E3%81%99%E3%82%8B-6c727329-d8fd-44fe-83b7-fa7fe3d8ac7a)