PDF 导出功能测试

完整测试版本，全面测试 PDF 导出工具的各种格式渲染效果

PDF Book Exporter

2025 年 08 月 06 日

目录

目录

第 1 章

说明

本书是专门用于测试 PDF 电子书导出功能的综合示例文档，包含了基础格式、代码表格、高级格式等各种 Markdown 内容，用于验证 PDF 导出工具的渲染效果。

1.1　测试目标

本测试文档涵盖以下内容：

1.1.1　基础格式测试

- ✅ 中英文混排显示
- ✅ Emoji 表情符号渲染
- ✅ 基础文本格式（粗体、斜体、删除线等）
- ✅ 列表和引用格式
- ✅ 链接和特殊字符

1.1.2　代码和表格测试

- ✅ 多语言代码块语法高亮
- ✅ 超宽代码块处理
- ✅ 各种表格格式
- ✅ 代码与表格混合内容

1.1.3　高级格式测试

- ✅ 数学公式渲染
- ✅ 任务列表和复选框
- ✅ 定义列表
- ✅ 复杂嵌套结构
- ✅ 特殊布局样式

开始全面测试！🚀

第 2 章

基础格式测试

2.1 文本格式测试

2.1.1 中英文混排测试

这是一段包含**中文粗体**和 English italic 的混合文本。我们来测试一下`inline code`的效果，以及删除线的显示。

Here’s some English text with **bold formatting** and italic formatting. Let’s also test `inline code` and ~~strikethrough~~ text.

在 Hugo 中，函数是在模板动作中使用的代码片段，它们接收一个或多个参数并返回一个值。与方法不同，函数不与特定的对象关联。Hugo 作为一个用 Go 语言编写的静态站点生成器，其函数体系充分利用了 Go 语言的特性，提供了高性能和丰富的功能。

只包含中文的段落：这部电影还有一个隐而不宣的主题：我们到底是在打自己的比赛，还是在西方的评分标准中争头名？郎朗在德国、仙台、柯蒂斯拿第一名；回国却被国内老师打成第三，说他“要摆正位置”；他用西方技法打败西方人，却始终无法获得完全的文化认同。

《Creep》是英国摇滚乐队 Radiohead 的首支单曲，发行于 1992 年，后收录于专辑《Pablo Honey》中。以下是其创作背景和表达含义：

- **创作背景**：根据乐队贝斯手 Colin Greenwood 回忆，这首歌由主唱 Thom Yorke 在大学时期创作。吉他手 Jonny Greenwood 曾表示，歌曲灵感来源于 Thom Yorke 喜欢的一个女孩，她突然出现在乐队的某次演出中。另有说法称，Thom Yorke 从未与那个女孩正式交谈过，只是偶尔在酒吧里看到她，当他终于喝得烂醉鼓起勇气表白时，她却被吓跑了。
- **表达含义**：这首歌主要表达了爱情中的自卑、自我怀疑以及对被接纳和理解的渴望。歌曲将喜欢的人比作“天使”，形容其如羽毛般轻盈美好，凸显出对方的完美与独特，而自己则是“a creep”“a weirdo”，觉得与周围世界格格不入，强烈的自卑感油然而生。歌词“I wish I was special, you’re so fuckin’special”体现了主人公对自身平凡的不满，渴望变得特别，能与心仪之人相配，却又深知彼此差距，充满了无奈与辛酸。同时，歌曲也反映了主人公内心的矛盾与挣扎，他渴望拥有完美的身

体和灵魂，希望能得到对方的注意，却又因自我认知而陷入痛苦，在爱情面前犹豫不决。整首歌通过主人公的内心独白，不仅是对个人情感的深刻剖析，也是对当代社会中人们在人际关系中常感到迷失和孤独这一现象的反思。

2.1.2 Emoji 表情符号测试 😊

- 基础表情：😀😃😄😁😆😅😂🤣
- 手势表情：👍👎👌🤝👋🙏💪✊
- 心形表情：❤️💙💚💛🧡💜🖤🤍
- 动物表情：🐶🐱🐭🐹🐰🦊🐻🐼
- 食物表情：🍎🍌🍇🍓🥝🍒🥑🥕
- 技术相关：💻📱⌨️🖥️🖨️📡🔌💾

2.2 列表格式测试

2.2.1 无序列表

- 第一级列表项
  - 第二级列表项
    - 第三级列表项
      - 第四级列表项
- 另一个第一级项目
  - 包含**粗体**的列表项
  - 包含斜体的列表项
  - 包含`代码`的列表项

2.2.2 有序列表

1. 第一个有序项目
   1. 嵌套的有序项目
   2. 另一个嵌套项目
      1. 更深层的嵌套
      2. 继续嵌套测试
2. 第二个主要项目
3. 第三个主要项目

2.2.3 混合列表

1. 有序列表开始
   - 嵌套的无序列表
   - 另一个无序项目
     1. 再次嵌套有序列表
     2. 继续有序项目
2. 回到主要有序列表

2.3 引用和分隔线测试

2.3.1 普通引用

> 这是一个普通的引用块。引用块通常用于突出显示重要的文本或者引用他人的话语。引用块通常用于突出显示重要的文本或者引用他人的话语。

2.3.2 包含格式的引用

> **重要提示**：这个引用块包含了多种格式
>
> - 列表项目 1
> - 列表项目 2
>
> `代码示例` 和斜体文本也可以在引用中使用。

2.3.3 分隔线测试

上面是一条分隔线，下面还有一条：

2.4 链接和图片测试

2.4.1 链接测试

- 普通链接：Google
- 中文链接：百度搜索

- 邮箱链接：rootsongjc@gmail.com
- 自动链接：https://github.com

2.4.2 图片测试（占位符）

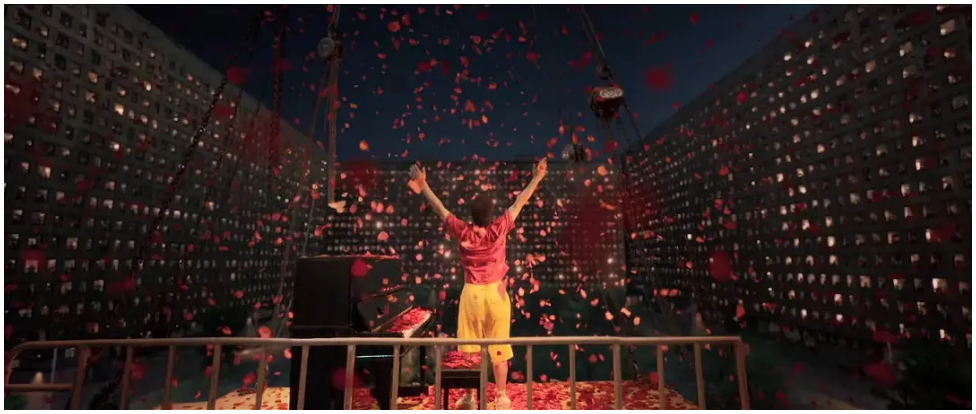

图 2-1: 示例图片

注意：实际 PDF 导出时，网络图片可能需要特殊处理

2.5 脚注测试

这里有一个脚注引用[1]，还有另一个脚注[2]。

脚注可以帮助提供额外的信息而不打断正文的流畅性[3]。

2.6 特殊字符测试

2.6.1 数学符号

α β γ δ ε ζ η θ ι κ λ μ ν ξ ο π ρ σ τ υ φ χ ψ ω

Σ ∏ ∫ ∂ ∇ ∞ ± × ÷ ⩽ ⩾ ≠ ≈ ∝ ∈ ∉ ⊂ ⊃ ∪ ∩

2.6.2 货币符号

$ € £ ¥ INR RUB ₩ ILS ₫ CRC BRC FRF ITL MILL NGN ESP PKR

[1]这是第一个脚注的内容

[2]这是第二个脚注，使用了自定义标识符

[3]第三个脚注，包含**格式化文本**和 `代码`

2.6.3 其他特殊符号

© ® ™ § ¶ † ‡ • ‰ ′ ″ ” ’ ※ !! ?? ?! !? ;

本章节完成了基础格式的测试，接下来的章节将测试更复杂的格式。

第 3 章

代码块和表格测试

本章节专门测试代码块、表格等复杂格式在 PDF 中的渲染效果。

3.1 代码块测试

3.1.1 内联代码

在文本中使用 `console.log()` 或者 `print()` 这样的内联代码。中文文本中的 `代码片段` 测试。

3.1.2 Python 代码块

```python
# Python 代码示例 - 数据处理
import pandas as pd
import numpy as np
from datetime import datetime

def process_data(filename):
    """
    处理 CSV 数据文件
    Args:
        filename (str): 文件名
    Returns:
        pd.DataFrame: 处理后的数据
    """
    # 读取数据
    df = pd.read_csv(filename, encoding='utf-8')

    # 数据清洗
    df = df.dropna()
    df['timestamp'] = pd.to_datetime(df['timestamp'])

    # 数据转换
    df['value'] = df['value'].astype(float)

    return df

# 使用示例
if __name__ == "__main__":
```

```
    data = process_data('sample.csv')
    print(f"处理了 {len(data)} 条记录 :smile:")
```

这是代码块下面的文字，用来说明代码的用途，比如 Java 代码中的 interface 的实现 theory。

3.1.3 JavaScript 代码块

```javascript
// JavaScript 代码示例 - React 组件
import React, { useState, useEffect } from 'react';
import axios from 'axios';

const DataFetcher = ({ apiUrl }) => {
  const [data, setData] = useState(null);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);

  useEffect(() => {
    const fetchData = async () => {
      try {
        setLoading(true);
        const response = await axios.get(apiUrl);
        setData(response.data);
      } catch (err) {
        setError(err.message);
      } finally {
        setLoading(false);
      }
    };

    fetchData();
  }, [apiUrl]);

  if (loading) return '加载中...';
  if (error) return `错误: ${error}`;

  return [
    '数据展示',
    JSON.stringify(data, null, 2)
  ];

export default DataFetcher;
```

这是代码块下方的文字，不应该与代码块重叠，并且要保证上方的代码块可以正确的分页。

弹性云服务器（Elastic Cloud Server）是一种可随时自助获取、可弹性伸缩的云服务器，可帮助您打造可靠、安全、灵活、高效的应用环境，确保服务持久稳定运行，提升运维效率。根据业务发展需要，您可以随时变更规格、切换操作系统、配置安全组规则或调整配额。除此之外，您还可以实时查看监控指标及审计日志，以便及时了解弹性云

服务器的健康状态。

3.1.4 超宽代码块测试

```
# 这是一个非常长的命令行示例，用来测试 PDF 中超宽代码块的处理效果
docker run -d --name my-container --restart=unless-stopped -p 8080:80 -v
  ↪ /host/path/to/data:/container/data -e ENV_VAR_1=value1 -e ENV_VAR_2=value2 -e ENV_VAR_3=value3
  ↪ --network=my-network --memory=2g --cpus=1.5 my-image:latest
```

3.2 表格测试

3.2.1 基础表格

姓名	年龄	职业	城市
张三	28	工程师	北京
李四	32	设计师	上海
王五	25	产品经理	深圳

3.2.2 包含格式的表格

功能	状态	优先级	负责人	备注
用户登录	✅ 完成	🔴 高	@张三	已上线
数据导出	🔄 进行中	🟡 中	@李四	预计下周完成
旧功能	❌ 废弃	🔵 低	-	不再维护
`API` 接口	⏳ 计划中	🟠 中	@王五	需求评审中

3.2.3 复杂表格（包含代码和链接）

技术栈	版本	用途	示例代码	文档链接
React	18.2.0	前端框架	`<Component />`	官方文档
Node.js	18.17.0	后端运行时	`require('express')`	Node.js
Python	3.11	数据处理	`import pandas`	Python.org
Docker	24.0	容器化，开发者友好，支持多平台，可以在 Linux、Windows 和 macOS 上运行，可以用 Orbstack 替代	`docker build .`	Docker Hub

3.2.4 包含内联代码和中文混排的表格

匹配方式	描述	示例
`prefix`	前缀必须与 `:path` 头的开头匹配	`/hello` 匹配 `/hello`、`/helloworld`、`/hello/v1`

匹配方式	描述	示例
`path`	路径必须与 `:path` 头完全匹配	`/hello` 只匹配 `/hello`，不匹配 `/helloworld`
`safe_regex`	使用正则表达式匹配 `:path` 头	`/\{}d{3}` 匹配三位数字路径
`connect_matcher`	只匹配 CONNECT 请求	用于 HTTP CONNECT 方法

本章节完成了代码块和表格的测试，下一章将测试更多高级格式。

第 4 章

高级格式和特殊内容测试

本章节测试更复杂的格式，包括数学公式、图表、复杂布局等。

4.1 数学公式测试

4.1.1 内联数学公式

在文本中，我们可以使用内联公式，比如 $E = mc^2$ 或者 $\pi \approx 3.14159$。

中文文本中的数学公式：圆的面积公式是 $A = \pi r^2$，其中 r 是半径。

4.1.2 块级数学公式

$$\int_{-\infty}^{\infty} e^{-x^2} dx = \sqrt{\pi}$$

$$\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$$

4.1.3 复杂数学公式

4.2 任务列表和复选框

4.2.1 项目进度跟踪

- [x] 需求分析 ✅
- [x] 技术选型 ✅
- [x] 架构设计 ✅
- [ ] 前端开发 🔄
    - [x] 页面设计
    - [x] 组件开发

- ☐ 接口对接
- ☐ 测试验证

☐ 后端开发 ⏳

- ☒ 数据库设计
- ☐ API 开发
- ☐ 业务逻辑
- ☐ 性能优化

☐ 测试阶段 ⏳

☐ 部署上线 ⏳

4.2.2 学习计划

☒ **第一周**：基础知识

- ☒ HTML/CSS 复习
- ☒ JavaScript ES6+
- ☒ React 基础

☐ **第二周**：进阶内容

- ☒ React Hooks
- ☐ 状态管理 (Redux/Zustand)
- ☐ 路由管理

☐ **第三周**：实战项目

- ☐ 项目搭建
- ☐ 功能开发
- ☐ 测试部署

4.3 定义列表

4.3.1 技术术语

API Application Programming Interface，应用程序编程接口，是不同软件组件之间通信的规范。

REST Representational State Transfer，表现层状态转换，是一种软件架构风格。

GraphQL 一种用于 API 的查询语言和运行时，由 Facebook 开发。

4.3.2 编程概念

函数式编程 一种编程范式，将计算视为数学函数的求值，避免状态变化和可变数据。
面向对象编程 基于”对象”概念的编程范式，对象包含数据（属性）和代码（方法）。

4.4 复杂嵌套结构

4.4.1 多层嵌套列表

1. **前端技术栈**
    1. **框架选择**
        - React
            - 优点：
                - 组件化开发
                - 虚拟 DOM 性能优化
                - 丰富的生态系统
            - 缺点：
                - 学习曲线陡峭
                - 需要额外的状态管理库
    2. **构建工具**
        - Webpack
        - Vite
        - Parcel
2. **后端技术栈**
    1. **语言选择**
        - Node.js
            - 优点：JavaScript 全栈
            - 缺点：单线程限制
        - Python
            - 优点：简洁易读
            - 缺点：性能相对较低

4.5 特殊布局测试

4.5.1 警告框样式

> ⚠️ **警告**
>
> 这是一个警告信息，用于提醒用户注意重要事项。

> ℹ️ **信息**
>
> 这是一个信息提示，用于提供额外的说明。

> ✅ **成功**
>
> 操作已成功完成！

> ❌ **错误**
>
> 发生了错误，请检查输入并重试。

4.5.2 键盘快捷键

- 复制：Ctrl + C
- 粘贴：Ctrl + V
- 撤销：Ctrl + Z
- 保存：Ctrl + S
- 查找：Ctrl + F

4.6 章节总结

本章节测试了以下高级格式：

- ✅ 数学公式（内联和块级）
- ✅ 任务列表和复选框
- ✅ 定义列表
- ✅ 复杂嵌套结构
- ✅ 特殊布局和样式

测试内容涵盖了 PDF 电子书可能遇到的各种复杂格式，为导出工具提供了全面的测试用例。

第 5 章

SourceHanMono 字体测试

本文档用于验证 SourceHanMono 字体在代码块中的应用效果。

5.1 Python 代码示例

```python
def hello_world():
    """
    一个简单的函数，展示 SourceHanMono 字体效果
    """
    print("Hello, 世界! ")  # 中文注释
    print("こんにちは、世界! ")  # 日文注释
    print("안녕하세요, 세계!")  # 韩文注释

    # 数字和符号测试
    numbers = [1, 2, 3, 4, 5]
    symbols = ['!', '@', '#', '$', '%', '^', '&', '*']

    return "测试成功"
```

5.2 Bash 脚本示例

```bash
#!/bin/bash
# 这是一个包含中文的脚本

echo "开始执行脚本..."
echo "正在处理中文文件名：测试文档.txt"

# 创建包含中文路径的目录
mkdir -p "./测试目录/子目录"
ls -la "./测试目录"

# 函数定义
function 显示消息() {
    echo "函数名也可以是中文：$1"
}

显示消息 "这是一个测试消息"
```

5.3 JavaScript 代码示例

```javascript
// JavaScript 中的中文变量和注释
const 问候语 = "你好，世界！";
const 数字列表 = [1, 2, 3, 4, 5];

function 显示问候(名字) {
    console.log(`${问候语} ${名字}`);
    // 这里展示中文字符在等宽字体中的效果
    console.log("中文字符测试：测试");
    console.log("English text: test");
    console.log("混合文本：mix 测试 test");
}

显示问候("张三");
```

5.4 内联代码测试

以下是内联代码的测试：

- Python 变量：`变量名 = "中文值"`
- 文件路径：`/home/用户/文档/测试文件.txt`
- 命令示例：`ls -la 中文目录`
- 混合内容：`hello 世界 test`

5.5 等宽字符对齐测试

```
ASCII 字符:  ABCDEFGHIJKLMNOPQRSTUVWXYZ
中文字符:   你好世界测试字体显示效果验证
日文字符:   こんにちはテストフォント
韩文字符:   안녕하세요테스트폰트
混合内容:   Test 测试テストテスト
```

5.6 代码注释多语言测试

```go
package main

import "fmt"

// 主函数 - 中文注释
// メイン関数 - 日文注释
// 메인 함수 - 韩文注释
```

```go
func main() {
    // 变量声明
    message := "多语言字体测试"

    fmt.Println(message)
    fmt.Println("Hello, 世界！")       // 英文 + 中文
    fmt.Println("こんにちは、世界！")   // 日文
    fmt.Println("안녕하세요, 세계!")   // 韩文
}
```

此文档将帮助验证 SourceHanMono 字体是否正确应用于所有代码块和内联代码。

「几米宋」微信公众号

jimmysong.io