中文测试文档

Typst 测试

2025-11-19

简介
这是一个测试文档，用于验证 Typst 能否正确渲染中文字符。

字体切换演示
本段使用 Source Han Serif SC（思源宋体）—— 带有装饰性笔画的衬线字体：

中文（Chinese）是世界上使用人数最多的语言之一。汉字是中文的书写系统。这是传统的衬线字体，笔画末端有装饰性的衬线。

本段切换到 Source Han Sans SC（思源黑体）—— 现代无衬线字体：

中文（Chinese）是世界上使用人数最多的语言之一。汉字是中文的书写系统。这是现代的无衬线字体，笔画简洁、线条均匀。

本段切换回 Source Han Serif SC（思源宋体）：

通过对比可以看到，宋体（Serif）和黑体（Sans）的视觉差异非常明显。

常用词汇
- 你好 (Hello)
- 世界 (World)
- 测试 (Test)
- 文档 (Document)

混合内容
这段文字包含中文和 English 混合内容，用于测试 mixed language rendering。

数学公式测试：$E = mc^2$

代码示例
Typst 支持语法高亮的代码块。

内联代码示例
行内代码可以用反引号包围，例如：`println!("Hello")` 或 `int main()`。

也可以指定语言：`Vec<String>` 或 `SELECT * FROM users`。

直接在文件中编写代码块
以下是直接在 `.typ` 文件中编写的代码块示例：

Python 示例

```python
def fibonacci(n: int) -> int:
    """计算斐波那契数列的第 n 项"""
    if n <= 1:
        return n
    return fibonacci(n - 1) + fibonacci(n - 2)

# 测试
for i in range(10):
    print(f"F({i}) = {fibonacci(i)}")
```

Rust 示例

```rust
fn main() {
    let message = "你好，世界！ ";
    println!("{}", message);

    // 使用迭代器
    let numbers: Vec<i32> = (1..=5).collect();
    let sum: i32 = numbers.iter().sum();
    println!("Sum: {}", sum);
}
```

JavaScript 示例

```javascript
// 异步函数示例
async function fetchData(url) {
    const response = await fetch(url);
    const data = await response.json();
    return data;
}

// 箭头函数
const greet = (name) => `Hello, ${name}!`;
console.log(greet("Typst"));
```

SQL 示例

```sql
-- 创建用户表
CREATE TABLE users (
    id SERIAL PRIMARY KEY,
    username VARCHAR(50) NOT NULL,
    email VARCHAR(100) UNIQUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- 查询活跃用户
SELECT username, email
FROM users
WHERE created_at > NOW() - INTERVAL '30 days'
ORDER BY created_at DESC;
```

从外部文件读取的代码块

以下代码块从外部文件读取：

Bash 脚本示例

```bash
#!/bin/bash

# 中文注释：打印欢迎信息
echo "你好，世界！"

# 函数定义
greet_user() {
    local name=$1
    echo "欢迎，$name！"
}

# 调用函数
greet_user "张三"

# 循环示例
for i in {1..5}; do
    echo "计数：$i"
done

# 条件判断
if [ -f "test.txt" ]; then
    echo "文件存在"
else
    echo "文件不存在"
fi
```

C++ 代码示例

```cpp
#include <iostream>
#include <string>
#include <vector>

// 中文注释：简单的 C++程序示例
class 学生 {
private:
    std::string 姓名;
    int 年龄;

public:
    // 构造函数
    学生(const std::string& name, int age)
        : 姓名(name), 年龄(age) {}

    // 打印信息
    void 打印信息() const {
        std::cout << "姓名：" << 姓名
                  << "，年龄：" << 年龄 << std::endl;
    }
};
```

```
int main() {
    // 创建学生对象
    std::vector<学生> 学生列表 = {
        学生("张三", 20),
        学生("李四", 22),
        学生("王五", 21)
    };

    // 遍历并打印
    std::cout << "学生信息列表: " << std::endl;
    for (const auto& s : 学生列表) {
        s.打印信息();
    }

    return 0;
}
```

Makefile 示例

```
.PHONY: build_docker compile

all: compile

build_docker:
  bash ./typst_toolchain/build.sh

compile:
  bash ./typst_toolchain/compile.sh test.typ
```

Dockerfile 示例

```
FROM alpine:latest

# Install dependencies
RUN apk add --no-cache \
    curl \
    ca-certificates \
    fontconfig \
    freetype \
    harfbuzz \
    graphite2 \
    libgcc \
    libstdc++ \
    unzip

# Download and install Source Han Sans fonts from Adobe
# Using Simplified Chinese (SC) variant - adjust URL for other variants (TC, JP,
KR)
RUN mkdir -p /usr/share/fonts/source-han-sans && \
    cd /usr/share/fonts/source-han-sans && \
    curl -L https://github.com/adobe-fonts/source-han-sans/releases/download/2.
004R/SourceHanSansSC.zip -o SourceHanSansSC.zip && \
```

```
    unzip -q SourceHanSansSC.zip && \
    rm SourceHanSansSC.zip

# Download and install Source Han Serif fonts from Adobe
# Using Simplified Chinese (SC) variant - adjust URL for other variants (TC, JP,
KR)
RUN mkdir -p /usr/share/fonts/source-han-serif && \
    cd /usr/share/fonts/source-han-serif && \
    curl -L https://github.com/adobe-fonts/source-han-serif/releases/download/2.
003R/09_SourceHanSerifSC.zip -o SourceHanSerifSC.zip && \
    unzip -q SourceHanSerifSC.zip && \
    rm SourceHanSerifSC.zip

# Update font cache and verify installation
RUN fc-cache -f -v && \
    echo "=== Installed Source Han fonts ===" && \
    fc-list | grep -i "source han" | head -10

# Download and install Typst
# Using the latest Typst release for x86_64 musl (Alpine compatible)
RUN TYPST_VERSION=v0.12.0 && \
    curl -fsSL https://github.com/typst/typst/releases/download/${TYPST_VERSION}/
typst-x86_64-unknown-linux-musl.tar.xz \
    | tar -xJ -C /tmp && \
    mv /tmp/typst-x86_64-unknown-linux-musl/typst /usr/local/bin/typst && \
    chmod +x /usr/local/bin/typst && \
    rm -rf /tmp/typst-x86_64-unknown-linux-musl && \
    typst --version

# Create working directory
WORKDIR /workspace

# Default command
CMD ["/bin/sh"]
```

Python 长代码示例

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

"""
学生管理系统 (Student Management System)
一个简洁的学生管理系统示例，用于测试 Typst 如何处理代码文件。
"""

import json
import os
from datetime import datetime
from typing import List, Dict, Optional
from dataclasses import dataclass, field, asdict


@dataclass
class 学生:
```

```
    """学生类 - 存储学生基本信息"""
    学号: str
    姓名: str
    性别: str
    年龄: int
    专业: str
    入学年份: int
    成绩: Dict[str, float] = field(default_factory=dict)

    def __post_init__(self):
        if self.年龄 < 0 or self.年龄 > 150:
            raise ValueError(f"年龄必须在0-150之间: {self.年龄}")
        if self.性别 not in ["男", "女", "其他"]:
            raise ValueError(f"性别必须是'男'、'女'或'其他': {self.性别}")

    def 添加成绩(self, 课程名称: str, 分数: float):
        if 分数 < 0 or 分数 > 100:
            raise ValueError(f"分数必须在0-100之间: {分数}")
        self.成绩[课程名称] = 分数

    def 计算平均分(self) -> float:
        return sum(self.成绩.values()) / len(self.成绩) if self.成绩 else 0.0

    def 获取等级(self) -> str:
        平均分 = self.计算平均分()
        if 平均分 >= 90: return "优秀"
        elif 平均分 >= 80: return "良好"
        elif 平均分 >= 70: return "中等"
        elif 平均分 >= 60: return "及格"
        else: return "不及格"


@dataclass
class 课程:
    """课程类 - 存储课程信息"""
    课程代码: str
    课程名称: str
    学分: int
    授课教师: str


class 学生管理系统:
    """学生管理系统主类"""

    def __init__(self, 数据文件: str = "students.json"):
        self.学生列表: List[学生] = []
        self.课程列表: List[课程] = []
        self.数据文件 = 数据文件
        self.加载数据()

    def 加载数据(self):
        if os.path.exists(self.数据文件):
            try:
                with open(self.数据文件, 'r', encoding='utf-8') as f:
                    数据 = json.load(f)
```

```
                    for 学生数据 in 数据.get('学生', []):
                        self.学生列表.append(学生(**学生数据))
            except Exception as e:
                print(f"加载数据时出错: {e}")

    def 保存数据(self):
        数据 = {
            '学生': [asdict(s) for s in self.学生列表],
            '保存时间': datetime.now().isoformat()
        }
        with open(self.数据文件, 'w', encoding='utf-8') as f:
            json.dump(数据, f, ensure_ascii=False, indent=2)

    def 添加学生(self, 学生对象: 学生) -> bool:
        if any(s.学号 == 学生对象.学号 for s in self.学生列表):
            return False
        self.学生列表.append(学生对象)
        return True

    def 删除学生(self, 学号: str) -> bool:
        for i, s in enumerate(self.学生列表):
            if s.学号 == 学号:
                self.学生列表.pop(i)
                return True
        return False

    def 查找学生(self, 学号: str) -> Optional[学生]:
        return next((s for s in self.学生列表 if s.学号 == 学号), None)

    def 按专业查找(self, 专业: str) -> List[学生]:
        return [s for s in self.学生列表 if s.专业 == 专业]

    def 统计信息(self) -> Dict:
        if not self.学生列表:
            return {}
        有成绩学生 = [s for s in self.学生列表 if s.成绩]
        平均分列表 = [s.计算平均分() for s in 有成绩学生]
        return {
            '总人数': len(self.学生列表),
            '最高分': max(平均分列表) if 平均分列表 else 0,
            '最低分': min(平均分列表) if 平均分列表 else 0,
            '平均分': sum(平均分列表) / len(平均分列表) if 平均分列表 else 0
        }


def 主程序():
    """主程序入口"""
    系统 = 学生管理系统("students.json")

    # 添加示例学生
    学生1 = 学生("2021001", "张三", "男", 20, "计算机科学", 2021)
    学生1.添加成绩("高等数学", 95)
    学生1.添加成绩("程序设计", 88)
    学生1.添加成绩("数据结构", 92)
    系统.添加学生(学生1)
```

```
    学生2 = 学生("2021002", "李四", "女", 19, "计算机科学", 2021)
    学生2.添加成绩("高等数学", 78)
    学生2.添加成绩("程序设计", 85)
    系统.添加学生(学生2)

    学生3 = 学生("2021003", "王五", "男", 21, "软件工程", 2021)
    学生3.添加成绩("高等数学", 92)
    学生3.添加成绩("程序设计", 95)
    系统.添加学生(学生3)

    # 添加课程
    系统.课程列表.append(课程("CS101", "程序设计基础", 4, "李教授"))

    # 显示统计和查找示例
    统计 = 系统.统计信息()
    print(f"学生总数: {统计['总人数']}, 平均分: {统计['平均分']:.2f}")

    if 学生 := 系统.查找学生("2021001"):
        print(f"找到学生: {学生.姓名}, 等级: {学生.获取等级()}")
    系统.保存数据()

if __name__ == "__main__":
    主程序()
```

结论

如果你能看到这些中文字符和不同的字体效果，说明 Typst 成功渲染了 Adobe Source Han 字体系列！