

# Slide 文字竖排实现

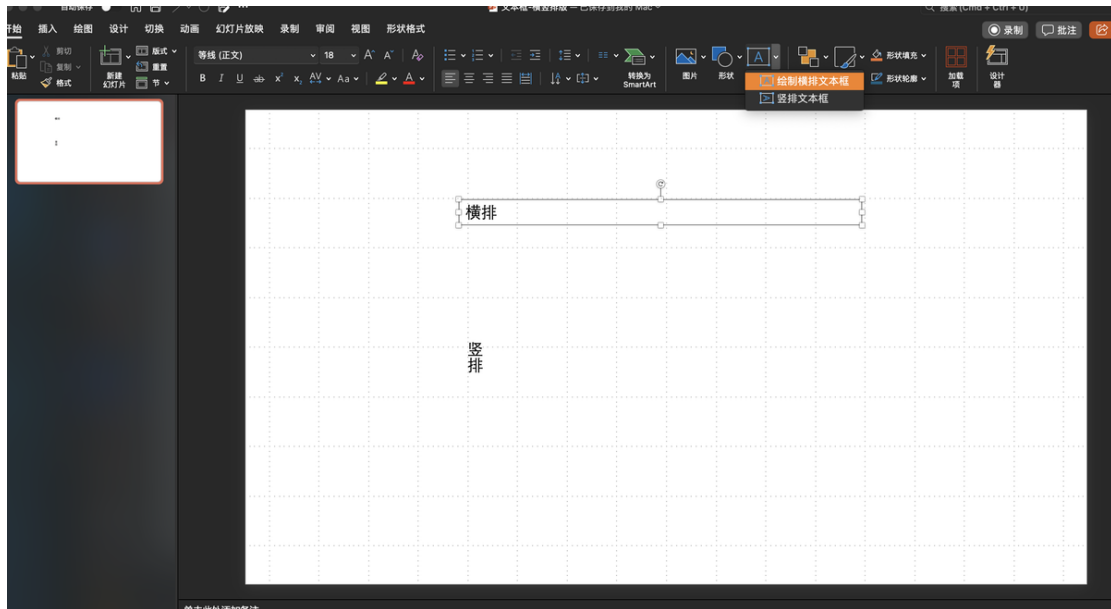
主要解释了：

- 1、文字横排竖排的格式解析
- 2、如何实现竖排渲染效果

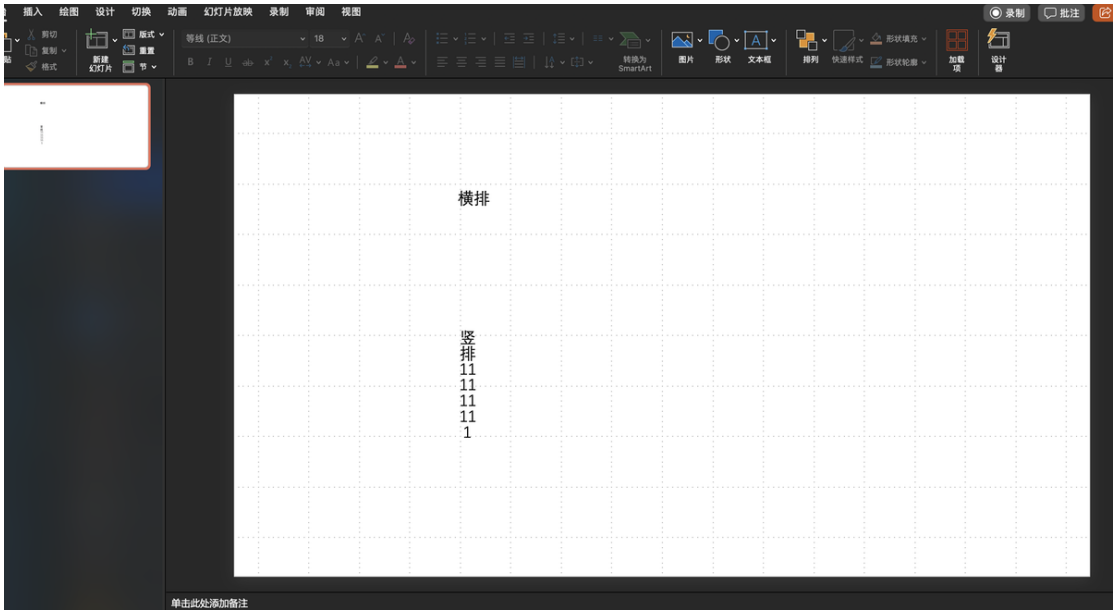
## 格式解析

### office 表现

- 1、设置入口如下



- 2、和腾讯文档的区别：竖排时占排：二个数字      一个中文字符



## OOXML 格式

默认显示横排，`vert="eaVert"`（强制竖排）。

```
<p:txBody>                                <!-- 文本框定义开始 -->
  <a:bodyPr vert="eaVert" ...>             <!-- 竖排+自动换行 -->
    <a:spAutoFit/>                          <!-- 自动调整高度 -->
  </a:bodyPr>
  <a:lstStyle/>                             <!-- 无自定义列表样式 -->
  <a:p>                                     <!-- 段落开始 -->
    <a:r>                                   <!-- 格式一致的文本段 -->
      <a:rPr kumimoji="1".../>              <!-- 中文+直排优化 -->
      <a:t>竖排</a:t>                      <!-- 文本内容 -->
    </a:r>
  </a:p>
</p:txBody>
```

### 横排竖排对比

属性/节点	当前代码（横排）	竖排版本（前一示例）
<a:bodyPr>	无 <code>vert</code> （默认 <code>horz</code> ）	<code>vert="eaVert"</code> （强制竖排）
文本方向	水平排列	从上到下竖排

自动换行	相同 ( wrap="square" )	相同
语言标记	相同 ( 中英文支持 )	相同

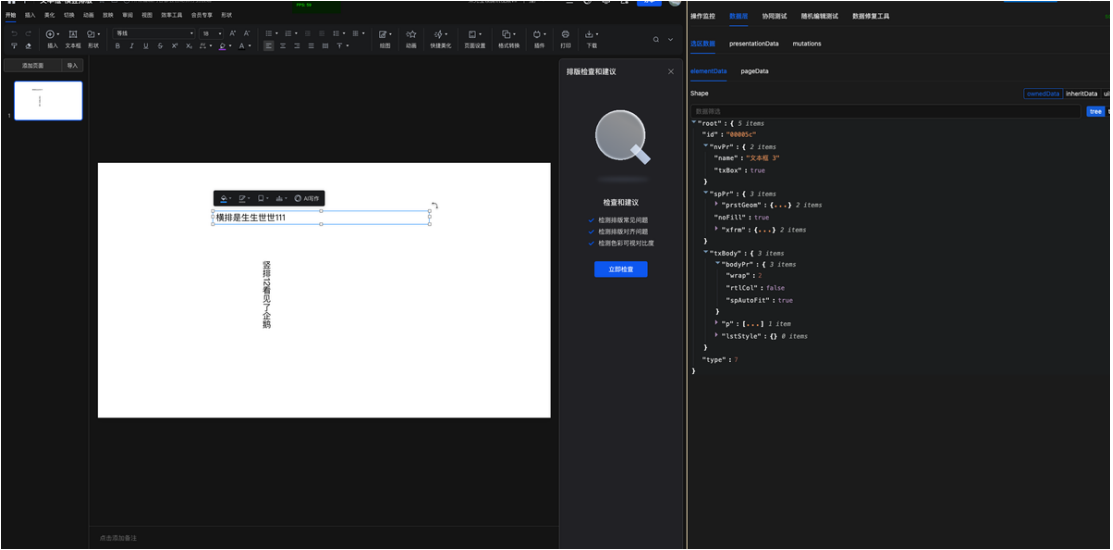
文本垂直排版方式枚举

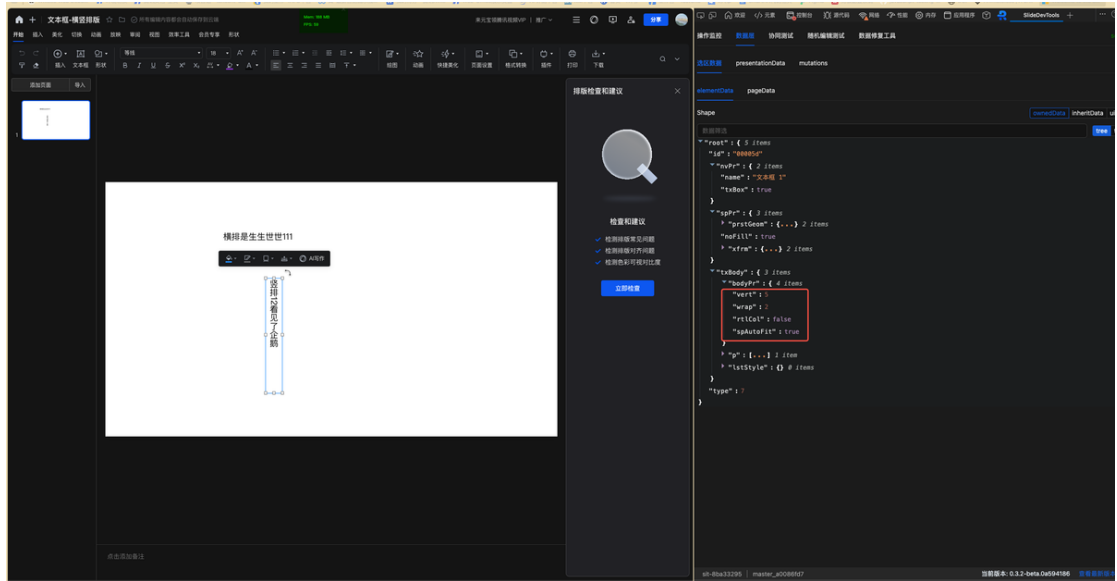
枚举值	值	含义
PBDEFAULT	0	默认值 ( 视上下文决定水平或垂直 )
horz	1	标准水平 ( 从左到右, 多行从上到下 )
vert	2	标准垂直 ( 从上到下, 多行从右到左 )
vert270	3	270° 旋转垂直 ( 从上到下, 多行从左到右 )
wordArtVert	4	WordArt 垂直 ( 单个字符竖排, 行从左到右 )
eaVert	5	东亚垂直 ( 中文/日文/韩文传统竖排, 从上到下, 行从右到左 )
mongolianVert	6	蒙古文垂直 ( 特殊竖排规则, 文字旋转 90° )
wordArtVertRtl	7	WordArt 垂直 ( 从右到左 ) 类似 wordArtVert ,但行方向相反 )

slide-sdk

sdkData 表现

可以看到竖排由 bodyPr.vert 属性控制, 且 vert 枚举值为 5 = eaVert , 对应东亚垂直。

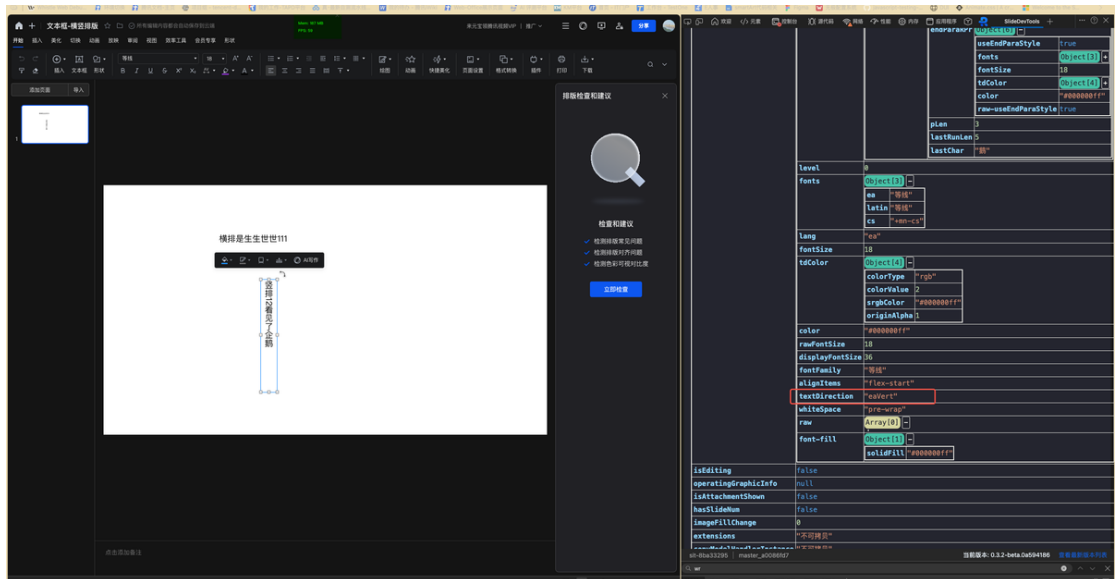




## 渲染解析

## UIData

对应的 uiData 为 **textDirection** , 如下。



转化函数如下

```
// 转换文本方向
export function convertTextDirection(sdk, style) {
  if (_has(sdk, 'vert')) {
    style.wrapperStyle = style.wrapperStyle || {};
    style.wrapperStyle.textDirection = sdk.vert;
  }

  return style;
}
```

canvas 渲染时对应的枚举，明显看得出来于文本垂直排版方式枚举对应。

```
export enum TextDirectionType {
  HORIZONTAL = 'horz',
  VERTICAL = 'eaVert',
  ROTATE_90 = 'vert',
  ROTATE_270 = 'vert270',
  WORD_ART_VERTICAL_RTL = 'wordArtVertRtl',
  WORD_ART_VERTICAL_LTR = 'wordArtVert',
  // 这个选项不是一个有效值，只是用于概括表示两种堆积方式
  WORD_ART_VERTICAL_GENERAL = 'wordArtVertGeneral',
}
```

## 渲染实现

### workbench 配置

主要是 controller 与 MENU\_SCHEMA 的配置。特别注意菜单栏的层级层级关系也配置在此。

```

MENU_SCHEMA
  83 import { ShowCommentAnchorController } from '@framework/workbench/ui/comment/show-comment-anchor/show-comment-anchor-controller';
  84 import { ShowCommentNotifyFlagController } from '@framework/workbench/ui/comment/show-comment-notify-flag/show-comment-notify-flag-controller';
  85 import { AddRevisionController } from '@framework/workbench/ui/add-revision/add-revision-controller';
  86 import { MachineCheckController } from '@src/framework/workbench/ui/quick-setting/machine-audit/machine-audit-controller';
  87 import { AMenuController } from '@framework/workbench/ui/titlebar/ai-menu/ai-menu-controller';
  88
  89 export default class TestWorkbenchSetting extends BasePCWorkbenchSetting {
  90   protected getController(): Function[] {
  91     return [-];
  92   }
  93
  94   /**
  95    * 菜单 schema, 子菜单项
  96    */
  97   @protected
  98   protected getMenuSchema(): WorkbenchMenuSchema {
  99     return MENU_SCHEMA;
  100   }
  101
  102   /**
  103    * 侧边栏 bar schema, 子菜单项
  104    */
  105   @protected
  106   protected getToolbarsSchema(): WorkbenchToolbarsSchema {
  107     return TOOLBAR_SCHEMA;
  108   }
  109
  110   /**
  111    * 顶部 titlebar schema, 子菜单项
  112    */
  113   @protected
  114   protected getTitlebarSchema(): WorkbenchTitlebarSchema {
  115     return Slide.TITLEBAR_SCHEMA;
  116   }
  117 }

```

```

src > framework > workbench > schema > menu-schema > common > index.js | @COMMON_INSERT_LIST
  83 export const INSERT_CHART = {
  85   items: [
  122     {
  123       key: 's(ActionType.INSERT_CHART)-radar',
  124       items: [
  125         chartConfig.RADAR_CHART_CONFIG.id,
  126         chartConfig.RADAR_WITH_POINT_CHART_CONFIG.id,
  127         chartConfig.RADAR_WITH_FILL_CHART_CONFIG.id,
  128       ],
  129     },
  130   ],
  131 };
  132
  133 // 插入内容
  134 export const COMMON_INSERT_LIST = [
  135   ActionType.AI_GENERATE_PAGE,
  136
  137   ActionType.INSERT_TEXT,
  138
  139   {
  140     key: 's(ActionType.INSERT_SHAPE)-parent',
  141     items: [
  142       's(ActionType.INSERT_SHAPE)-parent-custom',
  143     ],
  144   },
  145   ActionType.INSERT_LAYOUT,
  146   {
  147     key: ActionType.INSERT_PLACEHOLDER,
  148     items: [
  149       ActionType.INSERT_PLACEHOLDER_HORZ_CONTENT,
  150       ActionType.INSERT_PLACEHOLDER_VERT_CONTENT,
  151       ActionType.INSERT_PLACEHOLDER_HORZ_TEXT,
  152       ActionType.INSERT_PLACEHOLDER_VERT_TEXT,
  153       ActionType.INSERT_PLACEHOLDER_PICTURE,
  154       ActionType.INSERT_PLACEHOLDER_TABLE,
  155     ],
  156   },
  157 ];
  158
  159 // 插入内容
  160 export const COMMON_INSERT_LIST = [
  161   ActionType.AI_GENERATE_PAGE,
  162
  163   {
  164     key: ActionType.INSERT_TEXT,
  165     items: [
  166       ActionType.INSERT_TEXT_H,
  167       ActionType.INSERT_TEXT_V,
  168     ],
  169   },
  170
  171   {
  172     key: 's(ActionType.INSERT_SHAPE)-parent',
  173     items: [
  174       's(ActionType.INSERT_SHAPE)-parent-custom',
  175     ],
  176   },
  177   ActionType.INSERT_LAYOUT,
  178   {
  179     key: ActionType.INSERT_PLACEHOLDER,
  180     items: [
  181       ActionType.INSERT_PLACEHOLDER_HORZ_CONTENT,
  182       ActionType.INSERT_PLACEHOLDER_VERT_CONTENT,
  183       ActionType.INSERT_PLACEHOLDER_HORZ_TEXT,
  184       ActionType.INSERT_PLACEHOLDER_VERT_TEXT,
  185       ActionType.INSERT_PLACEHOLDER_PICTURE,
  186       ActionType.INSERT_PLACEHOLDER_TABLE,
  187     ],
  188   },
  189 ];

```

## 竖排属性配置

设置 isVertical 标识判断，通过插入时设置 textBodyStyle.wrapperStyle.textDirection === eaVert 去设置竖排效果。

```

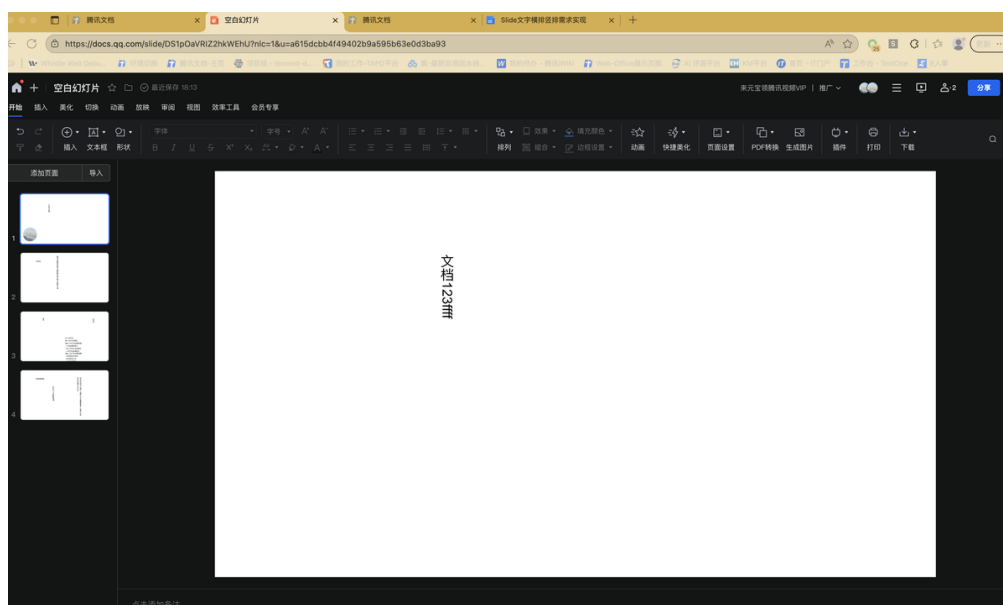
src > framework > editor > src > page > index > components > canvas > insert-element-wrapper > index.tsx > InsertElementWrapper > generateTextStyle
34 export default class InsertElementWrapper extends Component<InsertElementWrapperProps> {
137
138 /**
139  * 插入图形时设置文字默认水平居中和垂直居中
140  * 插入文本框设置默认格式为"根据文字调整形状大小"
141  * 插入文本框根据isVertical判断是否为垂直方向，默认为水平方向
142  */
143 private generateTextStyle(type: string, isAutoBreak?: boolean, isVertical?: boolean) {
144   if (type === InsertStateType.SHAPE) {
145     return createDefaultTextStyle();
146   }
147   if (type === InsertStateType.TEXTBOX) {
148     const textBodyStyle = {
149       autoFitType: AutoFitType.SHAPE_AUTO_FIT,
150       bodyStyle: isAutoBreak ? undefined : {
151         whiteSpace: WrapType.NO_WRAP,
152       },
153       ...(isVertical ? {
154         wrapperStyle: {
155           textDirection: TextDirectionType.VERTICAL,
156         },
157       } : {}),
158     };
159     return {
160       textBodyStyle,
161     };
162   }
163 }

```

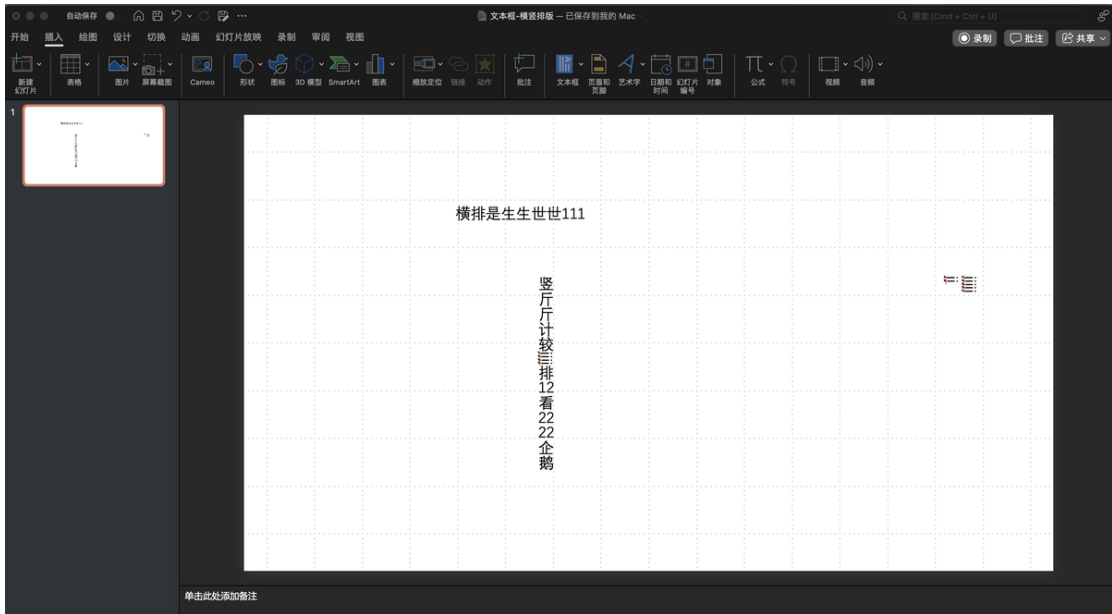
## 渲染差异

目前的竖排渲染与 office 有点区别，主要在数字渲染上，如下。

- 腾讯文档竖排时 数字的渲染会旋转 90 度，且每一排最多只能显示一个数字

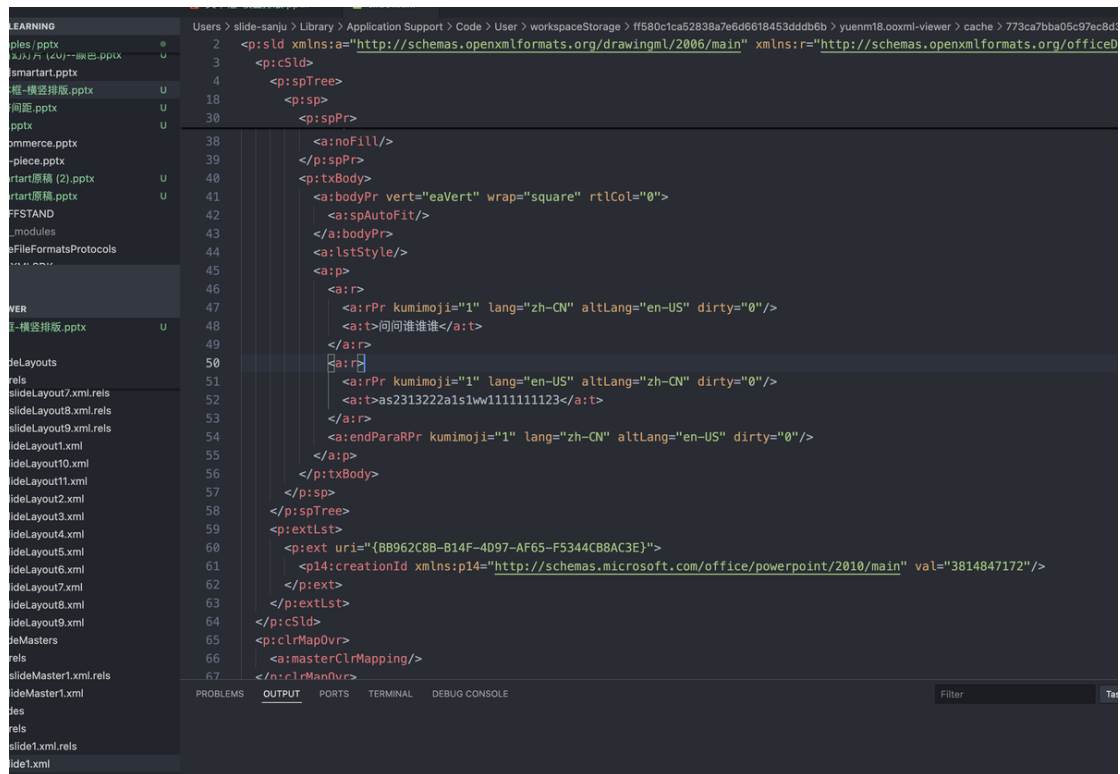


- office 会和字符一样正常渲染，且每一排最多可显示二个数字

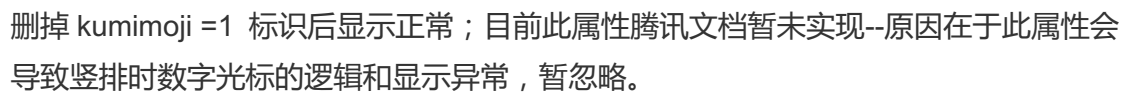


## 分析

发现是 office 的 bug ；如下图：将数字识别为中文了；且伴随光标显示有误







`kumimoji` 是 Office Open XML (OOXML) 中控制 **文本连字效果** 的属性，主要影响东亚文字（尤其是日文）的显示方式。

```
<XML>  
  
<a:rPr kumimoji="1" ... />
```

- ## 2. 作用效果

- **日文连字**：将相邻字符组合成更自然的字形（类似西文的 ligatures）。
  - 例：「はゝ」→可能显示为「はゝ」（连字效果）。
- **数字/字母间距调整**：改善密集文本的可读性。

属性名	取值	默认值	作用	适用场景
-----	----	-----	----	------

kumi moji	0	0	禁用连字效果，文本按普通方式显示。	普通西文或无需连字的 东亚文本
	1		启用连字效果（如日文“くみ文字”）， 优化相邻字符的组合显示。	日文排版、繁体中文竖排 或其他密集文本

## 2. 关联 XML 节点与属性

XML 节点或 属性	类 型	描述	与 kumimoji 的关联性
<a:rPr>	节 点	定义文本运行的格式属性( 如字体、 语言、连字 )。	直接包含 kumimoji 属性。
lang="en-US"	属 性	指定文本语言（如 en-US 英语， ja-JP 日语）。	日文语言下 kumimoji="1" 效果更显著。
vert="eaVert"	属 性	设置文本垂直排列（竖排），常见 于东亚语言。	竖排时连字可能影响行内对齐。
<a:spAutoFit/>	节 点	启用文本自动缩放以适应文本框。	若连字导致文本过长，可能触 发自动缩放。
<a:t>...</a:t>	节 点	包含实际文本内容( 如 1222222s )。	kumimoji 对其中字符生效 （尤其是日文）。

## 3. 效果对比示例

kumimoji 值	原始文 本	可能的显示效果（依赖字 体）	说明
0	は >	は > （普通分离显示）	无连字，字符独立。
1	は >	は > （可能连字为更紧凑形 式）	字体支持时会优化组合显示。
0	ffi	ffi （普通英文连字）	西文连字由字体控制，与此属性 无关。
1	ffi	i （部分东亚字体可能尝试 连字）	效果不保证，依赖字体设计。

## 4. 重要注意事项

项目	说明
字体依赖	必须使用支持连字的字体（如日文字体 MS Gothic/明朝）。
语言标记	结合 lang="ja-JP" 可增强效果（纯英文文本可能无变化）。
动态修改	通过 PowerPoint 开发者工具或直接编辑 XML 可调整此属性。
竖排兼容性	竖排模式（vert="eaVert"）下连字可能失效，需测试目标字体。

---