

# 集成和测试指南

## 替换文件步骤

### 1. 备份原文件



bash

```
cp CanvasTools.jsx CanvasTools.jsx.backup
```

```
cp PersonalSpace.jsx PersonalSpace.jsx.backup
```

### 2. 替换修复后的文件

- 用修复后的 `CanvasTools.jsx` 替换原文件
- 用修复后的 `PersonalSpace.jsx` 替换原文件

### 3. 验证导入

- 确保 `DraggableImage.jsx` 仍在同级目录
- 确保 `imageData.js` 仍在同级目录
- 确保 `index.js` 导出正确

## 功能测试

### 1. 绘画工具测试

#### 基本绘画



步骤：

1. 点击工具栏的"绘画工具"按钮
2. 工具按钮应该高亮显示
3. 鼠标光标变为十字或自定义笔刷
4. 在画布上拖拽鼠标画线条
5. 线条应该实时显示为青蓝色 (#61caff)

预期结果：

- ✓ 画线条流畅，无延迟
- ✓ 线条颜色为青蓝色
- ✓ 笔触圆滑（圆形端点）

### 持久化测试



步骤：

1. 完成一些绘画
2. 打开浏览器开发者工具 (F12)
3. 进入 Application > Local Storage
4. 查看 canvas\_draw\_paths 数据
5. 刷新页面 (Ctrl+R)
6. 检查绘画是否保留

预期结果：

- ✓ canvas\_draw\_paths 存储了路径数据
- ✓ 刷新后绘画保留
- ✓ localStorage 中包含 JSON 格式的路径数组

## 清除功能测试



步骤：

1. 完成一些绘画
2. 页面底部应该显示"清除绘画"按钮
3. 点击"清除绘画"按钮
4. 检查 localStorage 中的 canvas\_draw\_paths

预期结果：

- ✓ 按钮点击后所有绘画消失
- ✓ localStorage 中清除数据
- ✓ 再次刷新页面，绘画不会恢复

## 2. 文字工具测试

### 基本文字创建



步骤：

1. 点击工具栏的"文字工具"按钮
2. 鼠标光标变为文本光标 (I型)
3. 在画布上任意位置点击
4. 输入框应该出现在点击位置
5. 输入一些文字 (中文或英文)
6. 按 Enter 键确认

预期结果：

- ✓ 输入框自动聚焦
- ✓ 输入框有蓝色边框
- ✓ 按 Enter 后文字显示在画布上
- ✓ 输入框消失

## 文字拖拽



步骤：

1. 创建文字 (如上)
2. 鼠标 hover 到文字上方
3. 光标变为 grab (可抓取)
4. 按住鼠标左键拖拽文字
5. 松开鼠标

预期结果：

- ✓ 光标变为 grabbing
- ✓ 文字跟随鼠标移动
- ✓ 松开后文字固定在新位置
- ✓ 新位置保存到 localStorage

## 文字删除



步骤：

1. Hover 到已创建的文字上
2. 文字右侧应该显示"×"按钮
3. 点击"×"按钮

预期结果：

- ✓ 文字立即消失
- ✓ localStorage 中的 canvas\_text\_elements 更新
- ✓ 刷新页面后文字不恢复

## 文字输入处理



步骤：

1. 创建文字输入框
2. 输入中文：例如"测试文字"
3. 按 Enter 或点击其他地方确认

预期结果：

- ✓ 中文显示使用中文字体 (FZLanTingYuanS)
- ✓ 文字清晰可读
- ✓ 文字有白色背景，便于阅读

## 3. 套索工具测试

### 基本套索选择



步骤：

1. 点击工具栏的"套索工具"按钮
2. 鼠标光标变为十字
3. 在画布上画一个闭合路径，圈住一个或多个图片
4. 释放鼠标

预期结果：

- ✓ 套索路径显示为虚线
- ✓ 套索路径内的图片被选中（显示蓝色外发光）
- ✓ 图片 z-index 提升

## 精确选择测试



步骤：

1. 激活套索工具
2. 测试以下场景：
  - a) 圈住图片的部分边缘
  - b) 圈住图片的四个角
  - c) 圈住图片中心
  - d) 套索路径穿过图片但不闭合

预期结果：

- ✓ 所有场景中，只要套索与图片相交就会选中
- ✓ 多个图片可同时选中
- ✓ 选中状态清晰可见（蓝色发光效果）

## 多图片选择



步骤：

1. 激活套索工具
2. 画一个大范围的套索，包含多个图片
3. 检查选中状态

预期结果：

- ✓ 所有被圈住的图片都被选中
- ✓ 每个选中的图片都显示蓝色外发光
- ✓ 可以对多个选中的图片进行后续操作

## 4. 工具切换测试



步骤：

1. 依次点击三个工具按钮
2. 观察按钮的高亮状态和光标变化

预期结果：

- ✓ 只有一个工具按钮处于激活状态
- ✓ 再次点击已激活的按钮会关闭工具
- ✓ 光标随工具改变
- ✓ 只有激活的工具能进行操作

## 🔍 浏览器控制台检查

### 验证 localStorage



javascript

```
// 在浏览器控制台运行

// 查看所有保存的数据
console.log('Draw paths:', JSON.parse(localStorage.getItem('canvas_draw_paths')));
console.log('Text elements:', JSON.parse(localStorage.getItem('canvas_text_elements')));

// 清除所有数据 (仅测试用)
localStorage.removeItem('canvas_draw_paths');
localStorage.removeItem('canvas_text_elements');
```

## 验证事件监听



javascript

```
// 检查是否有错误
console.log('错误信息'); // 查看 console 标签

// 验证组件挂载
console.log(document.querySelector('.canvas-tools-overlay')); // 应该返回 SVG 元素
```

## 常见问题排查

### 问题 1：绘画不显示

症状：拖拽鼠标但画布上没有线条显示

排查步骤：

1. 检查工具是否激活（按钮是否高亮）
2. 检查 SVG 元素是否存在：



javascript

```
document.querySelector('.canvas-tools-overlay')
```

3. 检查是否有 JavaScript 错误 (F12 > Console)
4. 验证 canvas 元素的 ref 是否正确传递

解决方案：

- 确保 CanvasTools 组件被正确渲染

- 检查 canvasRef.current 是否有值
- 验证 activeTool 状态是否正确更新

## 问题 2：文字无法创建

症状：点击后没有输入框出现

排查步骤：

1. 确保文字工具已激活
2. 检查点击位置是否在 canvas 范围内
3. 查看浏览器控制台是否有错误

解决方案：

- 确保 canvas 有正确的宽高
- 检查 onMouseDown 事件是否被阻止
- 验证 currentText 状态是否更新

## 问题 3：套索选择不精确

症状：套索应该选中的图片没有被选中

排查步骤：

1. 在控制台验证碰撞检测：



javascript

```
// 检查图片的位置和大小
document.querySelectorAll('.canvas img').forEach(img => {
  const rect = img.getBoundingClientRect();
  console.log(img.id, rect);
});
```

2. 验证套索路径是否正确捕获
3. 检查图片位置计算是否考虑了 canvas 偏移

解决方案：

- 增加更多的检测点（可修改 corners 数组）
- 调整碰撞检测的灵敏度
- 验证 canvas 的绝对定位是否正确

## 问题 4：数据未持久化

症状：刷新页面后绘画或文字消失

## 排查步骤:

1. 检查 localStorage 是否有数据:



javascript

```
localStorage.getItem('canvas_draw_paths')
localStorage.getItem('canvas_text_elements')
```

2. 检查浏览器是否禁用 localStorage
3. 查看浏览器控制台是否有存储相关的错误

## 解决方案:

- 启用浏览器 localStorage (检查隐私设置)
- 检查存储空间是否已满
- 清除浏览器缓存后重试

## 性能监测

### 大量绘画时的性能



javascript

```
// 在控制台监测
performance.mark('draw-start');
// ... 进行大量绘画 ...
performance.mark('draw-end');
performance.measure('draw', 'draw-start', 'draw-end');
console.log(performance.getEntriesByName('draw')[0]);
```

## 优化建议:

- 如果有超过 100 条绘画路径，考虑使用 Canvas 而非 SVG
- 对 mousemove 事件进行防抖处理
- 使用 requestAnimationFrame 优化重绘

## 🚀 部署前检查清单

- 所有三个工具都能正常工作
- 数据能正确保存到 localStorage
- 刷新页面后数据恢复正确
- 没有 JavaScript 错误 (F12 Console)

- 图片选中和拖拽工作正常
- 工具按钮切换流畅
- 套索选择精确度满足需求
- 文字输入和拖拽工作正常
- 在不同分辨率下测试完成
- 在不同浏览器中测试完成

## 浏览器兼容性

已测试的浏览器：

- Chrome 90+
- Firefox 88+
- Safari 14+
- Edge 90+

特殊考虑：

- localStorage 在某些浏览器的隐私模式下可能不可用
- SVG 绘制在移动浏览器上可能有性能问题
- 触摸事件需要单独处理（当前代码仅支持鼠标）

## 获取帮助

如果遇到问题，请检查：

1. 浏览器控制台是否有错误信息
2. 是否正确替换了所有文件
3. 是否有文件导入问题
4. localStorage 是否可用