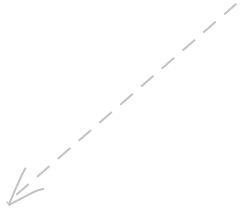
高效开发 & 安全开发

前端开发心得分享

点击空格访问下一页 →

高效开发

抽离公共逻辑，提高复用性，
进而实现高效开发



高效开发往往需要偷懒精神

高效开发

- 工具层面
  1. 用户代码片段
  2. VSCode插件
  3. NodeJS cli
  4. 个人工具包
  5. vite插件
  6. copilot & chatGPT

高效开发

- 代码层面
  1. 提高代码复用性
  2. 提高组件复用性
  3. 提高插件复用性
  4. 合理使用TypeScript

高效开发

通过预设语言或项目级别代码片段，快速生成代码

实测：手输15s，代码片段2s，效率比：7.5:1

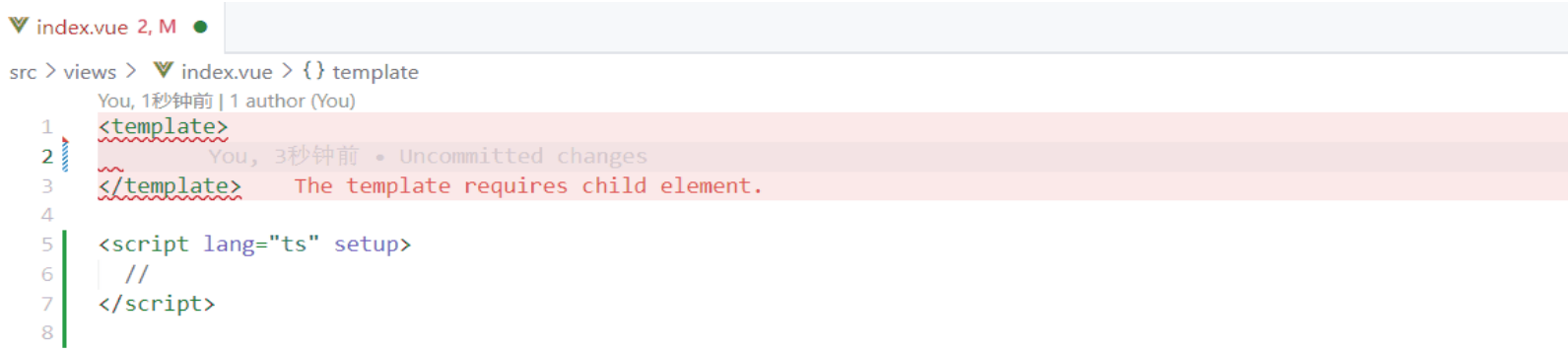
demo > 1-demo > index.vue > { } script setup

```vue
<template>
    <div>
        <h1>1-demo</h1>
    </div>
</template>

<script lang="ts" setup>

</script>
<style lang="less" scoped>

</style>
```

高效开发

通过vscode插件将开发表单简化为编写表单结构

实测：手输5min，代码片段10s，效率比：30:1



高效开发

通过NodeJS创建用户命令行交互界面（cli），简化开发场景，提高开发效率

- git-flow-cli，快速创建git分支，解决git分支命名和合并问题
- svn-deploy-plugin，快速将代码打包并部署到svn服务器，解决svn部署繁琐问题
- deploy-to-vm，快速将代码部署到虚拟机，解决手动打包部署问题
- link-manager，快速打开文件或目录，解决手动打开文件或目录问题

高效开发

通过维护个人工具包，在不同项目中快速复用代码

- 个人工具包
  - 通用组件，如：锁屏组件，无授权组件，导航菜单组件等等
  - 通用函数，如：cas号校验函数，有效期判断函数等等
  - 通用配置，如：axios配置，prettier配置，eslint配置等等
  - 通用模板，如：vue模板，cil模板，组件库模板等等

高效开发

通过AI辅助工具，提高开发效率

参考文档

高效开发

提高代码和组件的复用性，减少重复代码

按照局部级、项目级和平台级对代码进行抽离

- 函数
- 组件
- 插件

合理利用TypeScript的类型提示与类型检查

- 类型提示，提高代码编写效率
- 类型提示，提高代码重构效率
- 类型检查，减少手误
- 类型检查，在编译阶段发现代码问题

安全开发

前端开发常见的安全和体验问题

- CSRF
- XSS
- 禁用v-html
- 请求重复提交
- 字段安全校验
- 浏览器兼容性
- 响应性能优化
- 页面交互体验
- 页面按需加载

安全开发

CSRF，跨站请求伪造

用户已登录状态下，访问跨域接口，接口会认为是用户本人操作，从而执行接口操作。

CORS参考文档

注：由于ilabpower不是基于cookie验证用户登录状态，所以不存在CSRF问题，了解即可

安全开发

XSS，跨站脚本攻击

跨站脚本攻击（XSS）是一种常见的前端安全问题，攻击者可以在网页中注入恶意脚本，从而获取用户敏感信息或执行其他恶意行为。例如，在一个表单中输入恶意脚本代码，提交后该脚本就会被储存到数据库中并在下次页面加载时执行，导致用户信息泄露或者网站被劫持等问题。

防范措施：

1. 对输入进行过滤和转义，尤其是`<htmlTag xxx />`，`<script>`等标签，例如：`<script>alert(1)</script>`
2. 禁止使用`v-html`，使用`v-text`代替
3. 禁止内联脚本，使用 Content Security Policy，例如：`<meta http-equiv="Content-Security-Policy" content="script-src 'self'">`，只允许加载指定域名的脚本

禅道CIMS XSS bug列表

安全开发

- 数据去重提交
  - 前置拦截
  - 禁用按钮
  - 添加loading交互效果
  - 前端防抖

安全开发

- 字段安全校验，以swagger为准
  - 字段长度校验
  - 字段类型校验

安全开发

- 浏览器兼容性
  - 不同浏览器的CSS前缀：常见的CSS前缀有-webkit-、-moz-、-ms-、-o- 等
  - 使用ES5，ES6等标准语法
  - 针对特定浏览器增加兼容性代码

```
var buttonsContainer = document.getElementById('buttons-container');

if(buttonsContainer.addEventListener){
  // 其他浏览器
  buttonsContainer.addEventListener('click', function(event){
    // ...
  });
}else{
  // IE8
  buttonsContainer.attachEvent('onclick', function(event){
    // ...
  });
}
```

安全开发

- 响应性能优化
  - 减少HTTP请求，使用axios.all
  - 使用缓存，例如：浏览器缓存，CDN缓存减少服务器负载，提高访问速度
  - lazyload，按需加载，例如：图片懒加载，组件懒加载
  - 静态资源压缩，代码混淆，减少文件大小，提高访问速度
  - 异步加载脚本：使用 defer 或者 async 属性，让脚本在页面加载时异步加载，加快页面加载
  - 尽量减少dom操作，减少重绘和回流，提高页面性能

安全开发

- 页面交互和加载
  - 页面交互，loading交互，错误提示交互，成功提示交互，警告提示交互等等
  - 页面按需加载，路由按需加载，组件按需加载，图片按需加载等等
  - 页面加载，骨架屏，loading交互等等
  - 页面缓存，keep-alive缓存组件和路由等等