编号_____

本科生毕业设计

Web 前端组件智能生成系统

Web Front-end Component Intelligent Generation System

学生姓名	吴东岳
专业	软件工程
学号	170521325
指导教师	李松江
学院	计算机科学技术学院

二〇二一年六月

毕业设计（论文）原创承诺书

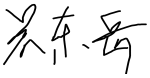
1．本人承诺：所呈交的毕业设计（论文）《Web 前端组件智能生成系统》，是认真学习理解学校的《长春理工大学本科毕业设计（论文）工作条例》后，在教师的指导下，保质保量独立地完成了任务书中规定的内容，不弄虚作假，不抄袭别人的工作内容。

2．本人在毕业设计（论文）中引用他人的观点和研究成果，均在文中加以注释或以参考文献形式列出，对本文的研究工作做出重要贡献的个人和集体均已在文中注明。

3．在毕业设计（论文）中对侵犯任何方面知识产权的行为，由本人承担相应的法律责任。

4．本人完全了解学校关于保存、使用毕业设计（论文）的规定，即：按照学校要求提交论文和相关材料的印刷本和电子版本；同意学校保留毕业设计（论文）的复印件和电子版本，允许被查阅和借阅；学校可以采用影印、缩印或其他复制手段保存毕业设计（论文），可以公布其中的全部或部分内容。

以上承诺的法律结果将完全由本人承担！

作 者 签 名：  　　2021年6月20日

摘要

本论文从课题研究背景着手，对前端开发的复杂度的现状进行了简单的论述，并且对课题的研究方法、研究内容进行了调研。接着对系统开发使用的相关技术理论进行介绍。本文设计实现的 Web 前端组件智能生成系统基于 Vue 框架，使用 ElementUI 作为组件库，VScode 作为开发平台，使用前端语言进行开发。系统需求分析部分，首先对整体需求进行了简单分析，接着分析了各模块需要实现的功能。系统详细设计和实现主要是对系统的功能模块进行详细设计并编程实现。最后系统经过对各模块的功能测试达到预期效果，所实现的功能基本满足了组件生成系统的工作需求。

关键字：Vue　Web 组件　智能生成　ElementUI

Abstract

This thesis starts from the research background of the subject, briefly discusses the current situation of the complexity of front-end development, and investigates the research methods and content of the subject. Then the related technical theories used in system development are introduced. The web front-end component intelligent generation system designed and implemented in this paper is based on the Vue framework, using ElementUI as the component library, VScode as the development platform, and the front-end language for development. In the system requirement analysis part, it first analyzes the overall requirements briefly, and then analyzes the functions that each module needs to implement. The detailed design and realization of the system are mainly to design and program the functional modules of the system in detail. Finally, the system achieves the expected results after functional testing of each module, and the realized functions basically meet the working requirements of the component generation system.

Key words：Vue;Web component; Intelligent generation; ElementUI

目录

第 1 章　绪论

1.1 研究目的和意义

近年来，各界开始重视组件化开发。组件开发是将 UI 样式的部分及其相应功能封装在一个单独的整体中，该整体被视为一个组件[1]。整个页面由大小组件连接而成，小组件既可以构成大组件，也是组成页面的基础。组件开发的优势在于可以提高代码的可读性和可维护性。除此以外，组件的功能明确分离，提高了组件的复用性，有利于组件之间相互组合和调用，提高开发效率。

目前为止，大多数现代前端框架，组件化都是最重要的基本概念之一，对于热门框架 Vue 和 React 等更是直接明显。一个完整的组件，一般会是由标记语言、业务逻辑以及样式一并构成，创建这类组件的目的是为了将杂乱的代码段创建为可重用的代码段[2]。

前端项目已经走向组件化，这个事实已经被大多数前端工程师所认识到。就像常规的面向对象语言中对类 class 的设计，为了设计一个低耦合、可复用的组件，需要考虑到各个方面，让它们能进行很好的复用和低耦合，这样的设计说起来容易做起来却很难，因为现实中往往没有足够的时间按照最优的方式去做。随着计算机技术普及，网络技术的发展使得越来越多的人能更方便地使用 Internet，人们也逐渐习惯科技给自己生活带来的改变，前端开发者更多利用网上的工具来简化日常的工作。因此构建基 Web 前端组件智能生成系统，在网络上提供更加方便快捷的服务。针对这种形势，开发一个以 Web 网站为平台，具有生成常用组件代码的系统，实现降低开发成本，是解决前端痛点的较好方案。

1.2 国内外研究现状

国外在组件化的研究和应用中投入了很多研究精力。国外前端领域主要有两个主流的框架，分别是 Angular 和 React。其中比较接近 Web Component 标准的框架，无疑是 Angular，而对于其他的框架，也正在朝标准靠拢[3]。作为较早的前端框架之一，Angular 是由谷歌公司开发的一个较成熟的开源框架。当时谷歌公司为了更方便的开发和维护单页面 Web App，开发了这个框架。在前端代码杂乱无章时，引入了 MVC 模式，让视图和数据逻辑分离，受到了用户群体的一致好评。有了充足的用户人数后，出现的问题也越来越多，且随着 ES 的升级，许多方法有了更好的实现。在这个基础上，Angular2.0 顺势推出，对比 Angular1.0，服务器端的预渲染明显加快，除此以外，编译速度也得到了很大的提升，检测变得灵活多样且敏捷，视图过渡也更新了不少细节，更加优雅高端。除此以外，推出了许多语法糖，让语法更加自然易懂，大大降低团队开发的复杂度。最重要的一点是，从 2.0 开始，原生的模块化 Web Component 概念被引入了 Angular 中，

取代了之前谷歌公司自己的组件概念。而 React 走的是另一条路，组件化并不是其考虑的重点。对于 React 来说，返回代码段才是他的核心。React 会将要渲染的界面作为一个整体组件，进行部分操作后，将返回值都放在 JS 中，这时的 JS 已经是个完整页面了，但 React 不止如此，他还会使用虚拟 DOM 技术进行渲染。用 JS 返回界面的方法，直接看着似乎并不理想，效率和其它以组件为中心的框架比起来较为低下。然而 facebook 也考虑到了这一点，于是创造了另一种提高效率的方法，使用虚拟 DOM 技术，对已有的 DOM 和将改变的 DOM 进行比较，找出它们变化了的地方，只对有变化的部分进行渲染，大大降低了渲染所需时间和性能。虽然在 React 中不能直接使用原生组件模块，也就是 Web Components，但只要经过一些简单的调整，还是能使用原生 Web Components 的。Web Components 在进行属性声明后，几个组件可以他们互相独立运行，也可以使用通信函数从而与其他组件进行交互，来得到其它组件的反馈。

国外的框架情况如上所述，而国内的框架中，最具有代表性的无疑是 Vue 框架。Vue 的开发者是前端圈耳熟能详的大佬尤雨溪，秉着数据驱动的原则，开发的一个非常优秀的前端框架。简单易用，它最明显的特点就是双向绑定，Vue 提供大量接口来实现数据双向绑定，对于无法理解原理的新手十分友好，只需要直接调用组件即可。Vue 的核心思想是对数据进行双向绑定与组件化，数据双向绑定的效果是让数据能直接渲染到视图上，DOM 元素和数据在这种情况下就可以不需要考虑同步问题。Vue 中另一个重要的部分是组件化，大部分代码将以组件的方式出现在项目中。

1.3 研究方法与研究内容

本系统主要目标是根据前端开发人员常用的组件，以软件工程的思想设计一个完整的 Web 组件智能生成系统：开发者可以通过拖拽，填写表单等方式进行组件的设计，最后使用生成按钮获得代码或是代码文件。本系统在 Vue.js 框架的基础上进行组件化的开发，前端使用了最契合 Vue 的组件库 ElementUI 框架完成页面设计，VScode 作为开发平台。本文遵循了常规的系统开发方式，设计并实现了 Web 前端组件智能生成系统，本文的组织结构如下：

第一章系统的研究背景与意义、国内外研究现状，并简述本课题的研究方法及内容。第二章介绍了系统开发的相关理论，主要介绍系统的相关技术，包括 Vue.js 框架、ElementUI 框架、和拖拽生成技术。第三章为系统需求分析部分。主要介绍 Web 前端组件智能生成系统需要实现哪些功能，说明本系统的框架以及功能模块。第四章为系统详细设计部分，针对 Web 前端组件智能生成系统设计进行详细说明。第五章为系统实现与测试部分，对本文设计实现的 Web 前端组件智能生成系统进行项目部署和功能测试。第六章为总结和展望，对本文设计实现的系统所做的工作归纳总结，并且对未来的目标进行展望。

第 2 章　理论基础

2.1 前端基础

2.1.1 HTML

HTML 中文名为“超文本标记语言”，从 1990 年 HTML 标准被确定以来，它就一直作为互联网中的信息标识语言并一直沿用至今，中间标准几次轮换，形成了今天的较为完善的结构，HTML 书写的内容需要由浏览器进行翻译并显示，网页的本质是由超文本标记语言组成的页面,加上一些其他技术如 JS 和 CSS 等，即可创建出内容丰富的网页。

HTML 书写相对容易，而且功能强大，能够将不同格式的数据嵌入网页中，譬如图片、音频、视频等。其具有简易性、可拓展性、平台无关性以及通用性的特点，HTML 不在乎用户使用的是什么系统什么浏览器，只要是使用能够正常工作的浏览器，HTML 里边所包含的内容即可在用户面前展开，最终渲染出 DOM 树。

2.1.2 JavaScript

JavaScript 通常简称为 JS，是前端三件套中最重要的一项技术。不同于其他较为庞大面向对象语言，JS 是一种轻量级的编程语言，会进行即时的编译。它有着编写简单，容易上手，快速反馈的特点。随着 JS 的用户越来越多，它甚至被用到了很多非浏览器中，如出名的 Node.js 编译器就是使用 V8 的内核，让其可以在后端语言中使用 JS 语言。JavaScript 灵活多变，不仅能基于原型编程，在一定的调整后甚至支持面向对象式风格。JavaScript 是网景公司在 1995 年创造出来的，由于用户基数非常大，所以有了专门的标准 ECMAScript。2012 年后，基本所有浏览器都能够支持 ECMAScript。JavaScrip 具有以下特点:

不同于面向对象的语言，往往先编译后执行。当程序的运行时，JavaScript 才一行一行对代码读取并执行，他是一种解释型的脚本语言。JavaScript 对不会对使用的数据类型做出严格要求，他是弱类型的脚本语言，在运行时才对数据类型做出判断。其设计思路基本类似于 Java 的语句，相对更好上手。另外，JavaScript 它不必对 Web 服务器进行访问，只需要鼠标的点击、移动等操作就可以做出自己的响应。它并不依赖于服务器或是电脑系统，而是依赖于浏览器。JavaScript 几乎被所有浏览器支持，所以凡是可以使用浏览器的系统几乎都可以使用 JS。

2.2 Vue.js 框架

Vue.js 是一个由中国人开发的前端框架，它的核心是数据驱动，将所有的数据和视图进行绑定，我们只需要关心数据的逻辑，对于新手来说十分友好。它有两大特性：双向绑定和组件化。

双向绑定：Vue 的核心思想是对数据进行双向绑定，让数据能直接渲染到视

图上，DOM 元素和数据可以紧紧绑定在一起。过去数据响应时，往往要使用 Jquery，首先要去获取到想要更新的节点，然后将值赋给它，这部分操作相当繁琐，而且每次进行数据变化时，都需要进行该操作，让人烦不胜烦。如今通过 Vue 的双向绑定，当对数据进行操作时，由于有双向绑定的存在，我们不需要再去繁琐的写选择器，只需要保证业务逻辑是正确的即可，数据将会自动更新过去。因此代码能有更高的可读性并且降低维护成本。

组件化：在 Vue 中组件也是一个非常重要的部分，大部分代码将以组件的方式出现在项目中。组件的好处在于可以将经常使用的代码段分离出来，经过配置后成为可复用的组件，通过一个个组件来编写项目，有易维护、易复用的好处。同时多人开发中，也能划清各个使用者的工作文件。

2.3 MVVM 模式

Web 组件智能生成系统采用 Vue.js 的 MVVM 系统架构，与传统的 MVC 有些不同。其中 M 代表 Model，也就是模型，指从后端传递的数据，几乎所有的模型数据都会被放在 Model 层中。V 代表 View，也就是视图，指所看到的页面，该层的作用是将页面展示给外部。C 代表 Controller，也就是控制器，指具体的页面业务逻辑，繁琐的业务逻辑会移交到这里。使用 MVC 的目的就是将模型数据和页面的代码完全分离，通信的细节全部交由 Controller 来处理[4]。

也就是说，传统的 MVC 开发方式，需要通过触发视图层的指定事件，才能将数据传到 Controller 层，这时 Controller 才会去访问 Model 层获取相应数据，再返回给视图层进行渲染，显然是费时费力的。而对于 MVVM 架构来说，它自动同步了视图和数据，模型数据发生更新时，不必手动操作 DOM 元素，视图层的数据显示会自动进行更新。

MVVM 的出现后，大部分项目都开始分离前端开发与后端业务的逻辑，只考虑前端部分，降低了代码的理解成本，大大增加前端开发效率。MVVM 中的 M 是模型，V 是视图，和 MVC 近似，不同的地方在于 ViewModel 层。该层除了为模型数据进行转化，更是会双向数据绑定视图层进行数据交互。具体如图 2.1 所示。

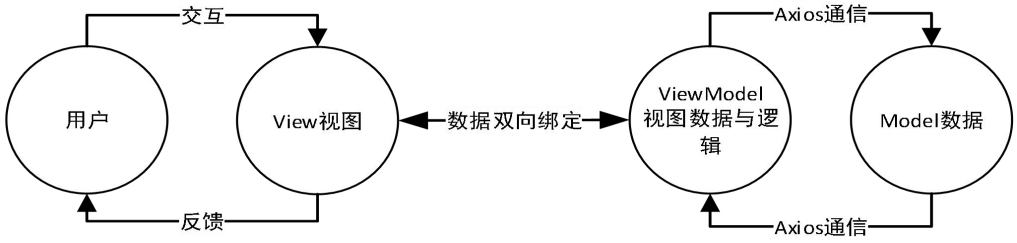


图 2.1 MVVM 模式结构图

2.4 拖拽模块

在网页中，拖拽并不少见，经常会有人将文字一段一段的拖入输入框。这就

是最基本的拖拽事件。拖拽的第一步是选中网页的一部分，紧接着需要按住鼠标，此时就可以对选中的目标进行拖动了。然而，这只是对于网页默认拖拽行为产生的特定效果，如果不进行监听程序拖拽数据和配置相应事件，除了图像、链接和选择文本这三种可以之外，其他组件是不可以进行拖拽的。

如果想要让 HTML 元素能够成为可拖拽目标。需要进行以下配置：第一步是将想要拖拽的元素的 draggable 的布尔值设置为真，第二步是监听 dragstart 事件，最后一步则是设置拖拽数据[5]。

属性 draggable 设置为真时，这个元素会变成可拖拽的。如果该属性被省略或被设置为假，则该元素将不可拖拽，此时拖拽只会选中文本。draggable 属性可在任意元素上设置，包括图像和链接。然而，对于后两者，该属性的默认值是 true，所以你只会在禁用这二者的拖拽时使用到 draggable 属性，将其设置为 false。拖拽所触发的事件如表 2.1 所示。

表 2.1 拖拽事件表

事件名称	描述
dragstart	在拖拽刚开始时触发。它主要用于将拖拽的组件信息传递给画布。
drop	在拖拽完全结束时触发。主要用于接收拖拽的组件信息。
mousedown	在组件上按下鼠标时触发。
mouseup	鼠标抬起时触发。
dragend	拖拽动作结束时触发。
dragover	鼠标置于组件上时触发。

第 3 章　需求分析与系统设计

3.1 需求分析

系统需求分析通过调查用户的需求，分析用户的需求，从而确定系统的主要功能，以及根据收集的用户需求，确定收集的数据分类和业务流程。系统需求分析作为系统程序设计的起点，在整个软件开发的生命周期中有着重要意义。

互联网行业快速发展，而网页又是最常见的开发工作，越来越多的前端需求，使得开发者在前端开发时面对频频出现的代码段难以复用，需要花费大量精力重写，更是增加了维护成本。

针对这种情况，需要设计一个专门的系统，来减少开发的难度。对于如何处理重复的代码段，业界已经有了成熟的方案——组件化开发[6]。通过 Vue 框架生成组件的能力，来进行代码的复用。但仅仅如此还不够，即使有了组件化开发，开发人员也要重复书写一些类似但又不可复用的常用组件，如表单、表格等。这些的编写也要耗费大量时间，所以此系统应当更进一步，将一些基础组件进行封装，填入部分属性后即可生成完整的组件。

对于 Web 前端组件智能生成系统，大致有以下设想：能够使用拖拽、填写表单等交互性操作来进行表单的设计，此时的组件库需要足够多且常用；能够在设计完之后，向用户展示设计的效果图；能将设计图完美的转化成组件，最好可以直接进行代码的预览和编辑。

3.2 功能需求分析

本系统是 Web 前端组件智能生成系统，系统的用户主要是前端开发者，因此在系统实现的过程中，应该充分的考虑前端开发者的需求。通过前期调研以及查询相关资料，分　析发现用户的功能需求大致为布局清晰，有较多适配的组件，导出组件简单，无需登录即可使用，从而节约时间。开发者在使用本系统生成组件时，需要能够配置大部分常用的组件及其属性。

基于以上分析，可列出以下基本需求：需要有一个较为简洁易用的页面来展示系统。该系统应有一个组件面板，存放常用的前端组件，该组件面板内的组件可拖拽也可点击。同时也应该存在负责渲染的画板，无论双击组件或是拖拽组件都可以使其呈现于画板上。当绘制完基础组件后，可以生成组件代码，并提供一个导出文件的按钮。

下面是对组件生成系统的具体功能进行分析，如图 3.1 所示。

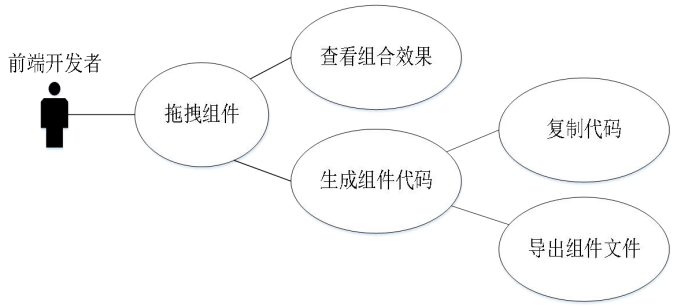


图 3.1 功能用例图

1. 拖拽组件：

使用该系统的工作人员登陆到组件生成系统后，页面的左侧为组件区域，可以使用鼠标对其进行拖拽，将其拖到预览区，松开鼠标即可绘制，无论后续执行任意操作，其基础都是将组件拖拽到预览区。只有构建好组件，才能继续下一步。

2. 查看组合效果：

可以直接看到绘制后的组件的效果，也可以拖动预览区的组件重排序其顺序，或者是删除、编辑已存在的组件，再查看其效果。

3. 生成组件代码：

按照渲染区的组件顺序，依次转化出组件代码，最后将组件代码组合到一起形成完整的组件。生成后，系统可以将生成的代码进行选取复制，或是使用按钮进行整体文件复制。考虑到设计的美观性，应当使用较简单的编辑器来进行展示，但编辑器组件需要考虑的地方较多，可以从 npm 上寻找到现成的模块；也可以将生成的代码直接进行文件导出，点击导出按钮后，会自动启动下载，将一个完整的 Vue 文件整体下载下来。

使用用例表对系统主要实现的功能点进行分析，组件智能生成用例表如表 3.1 所示。

表 3.1 组件智能生成用例表

名称	组件生成
系统用户角色	游客
前置条件	用户进入系统
主要路径	（1）清空样例组件 （2）将想使用的组件拖拽至预览区 （3）配置组件属性
结果	导出获得相应的组件代码

组件智能生成是该系统最重要组成部分。开发者在进入系统后，对所需要的

组件进行拖拽并生成，主要包括添加、删除、修改和导出组件。该系统拥有较多的常用组件，组件表如表 3.2 和 3.3 所示。

表 3.2 输入型组件表

名称	描述
单行文本	可输入单行文本
多行文本	可输入多行文本
密码	可输入不可见文本
计数器	仅能输入数字
编辑器	富文本编辑器

表 3.3 选择性组件表

名称	描述
下拉选择	可进行下拉选择，对应 input-select
级联选择	可进行多级下拉选择，下拉选择的详细版
单选框组	可从多个选项中选择一个，对应 input-radio
多选框组	可从多个选项中选择多个，对应 input-checkbox
开关	可选择 true/false
滑块	可选择百分比
时间选择/时间范围	可选择时间
日期选择/日期范围	可选择日期
评分	可选择固定数量
颜色选择	可选择颜色
上传	可进行上传文件操作

3.3 技术可行性分析

近几年前端方向在各种便利框架工具方面发展十分迅速，主流的前端框架也非常成熟，在预测时间和预测精度可以满足实时组件生成系统的设计。

框架上来看，本文所选用的 Vue.js 框架已经开源，其优势在于：比起其他框架的文档，Vue 的官方文档是简单易懂的；更新 DOM 节点时，使用了异步操作，能加快渲染速度；用组件化的方式来书写项目；仅仅只有 1.8kb 大小且无依赖；表达式和无需声明依赖的可推导属性；可以通过 NPM、Bower 或 Duo 安装，不强迫你所有的代码都遵循 Angular 的各种规定，使用场景更加灵活[7]。

组件库上来看，ElementUI 是基于 Vue 的组件库，美观而简洁。由于基于 Vue，学习成本更低，只需掌握 Vue 即可使用。而其组件库也收录在 npm 模块中，可以轻松的安装使用[8]。有了该组件，便省去了大部分构建美观组件的时间，并且生成的组件也是使用 ElementUI，使用相同的组件库可以减少学习新组件库所花费的时间。

再看其他模块部分，拖拽部分 JavaScript 有原生的 drag 属性支持，完全可以自己实现。导出文件部分和编辑器部分可以使用现有的成熟模块，配置部分属性即可。最重要的生成代码模块，可以使用组件化思想来完成。所以只需设计一个简单的系统界面，左侧组件区域，中间预览区域，右侧属性区域，最后留出一个生成按钮的工具栏即可满足要求，实验代码由 JavaScript 语言进行编写，使用 HTML、CSS 实现前台界面的展示，所以该系统在技术上满足开发的要求。

3.4 系统功能模块设计

拖拽绘制部分主要分为拖拽模块，属性模块，渲染模块 3 个模块。拖拽模块。在左侧的组件区域选中希望加入的组件，将其拉到渲染区域即可。属性模块，将预览区域的组件点击后，即可在右侧的属性区域配置该模块的各项基本属性。渲染模块，当组件以及其属性都配置完成时，中间的预览区域的组件也会随之变化，由于配置了双向绑定的功能，所以不需要进一步手动刷新。

智能生成部分分为：生成模块，编辑器模块，导出文件模块。生成模块是负责将预览区的可视组件转化为代码并输出[9]。编辑器模块是负责生成代码后，将代码直接展示给用户，而导出文件模块则是要将生成的代码写入文件中，并提供给用户下载。全部模块如图 3.2 所示。

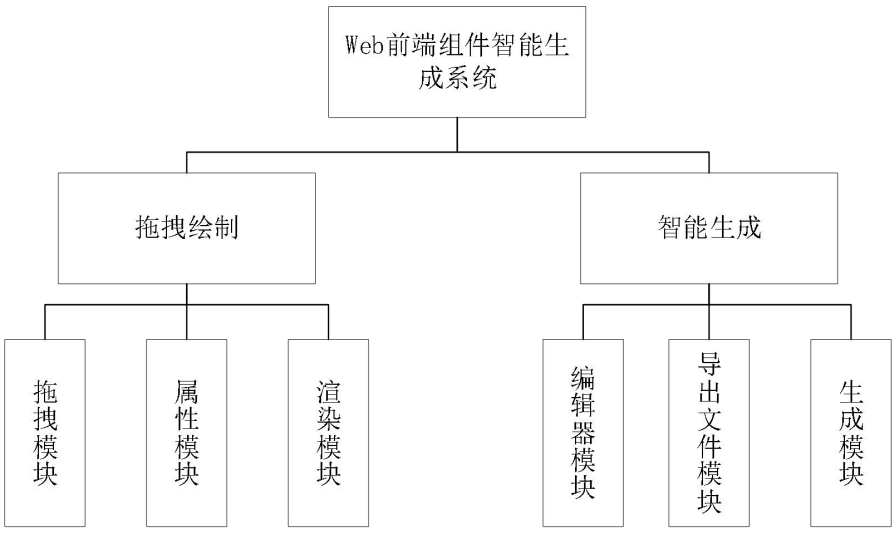


图 3.2 系统功能模块图

（1）拖拽模块：在左侧的组件区域选中希望加入的组件，将其拉到渲染区域即可，此外，渲染区域的组件也可以使用拖拽来进行位置交换等。拖拽的原理是为 DOM 节点绑定 draggable 属性以及相应的 ondragstart()、ondragover()事件，但本次的拖拽模块涉及大量节点，一个个绑定显然是不现实的，这里采用了设计模式的装饰者模式，将所有节点通过一个 Vue 组件处理，使其批量获得拖拽能力，降低了代码复杂度和耦合性。

此处没有使用相关的现成组件，而是使用原生方法自己封装，这样做的好处

就是能最大程度的利用灵活性，保证各种情况下能定位到问题所在，为了实现这一功能，使用系统原生的 draable 属性，为组件附加拖拽能力。从而降低了代码的理解难度，并且能够将更深入的理解拖拽的特性。

（2）属性模块：预览区域的组件点击后，即可在右侧的属性区域配置该模块的各项基本属性，不同的组件可配置的属性不一样。当组件被拖入预览区时，开发者在属性配置模块输入想要配置的属性，然后当鼠标从输入框中离开并失焦时。数据将利用双向绑定直接同步给当前预览区的组件。

这里的双向绑定使用的自然是 Vue 的技术，大大降低了手动修改 DOM 的成本。对于不同种类型组件会有不同种属性，这些属性统一配置在 config.js 文件中，一并进行读取并渲染出属性模板。

（3）渲染模块：当组件以及其属性都配置完成时，中间的预览区域的组件也会随之变化，由于配置了双向绑定的功能，所以不需要进一步手动刷新。由于渲染的组件需要保证一定的美观性，所以使用 ElementUI 进行渲染[10]。注意此时的渲染只是提供展示，和之后的生成组件的代码并不完全相同，起一个参考作用。在渲染模块中，依然可以进行拖拽操作，可以进行组件顺序的调整，对于不需要或是误操作的组件，也可以进行删除操作[11]。渲染时需要考虑顺序问题，

（4）生成模块：该处的组件会按顺序读取所有的组件，依次识别每个组件，判断出类型后，将组件上的属性注入该类型模板，未配置的属性会取默认值或为空，所有组件转化为代码后，形成真正的组件（字符串形式），最后将所有组件按顺序合并到一起，并装入容器中，成为一个完整的组件代码[12]。此处的代码是核心，是将可视化的组件转化为代码的重要过程。

（5）编辑器模块和导出文件模块：编辑器模块需要将生成模块的所有代码进行展示，可以进行局部代码的选取复制。有一个美观的编辑器相对重要，ElementUI 并没有提供编辑器的组件，故需要在 npm 中找到一个成熟的编辑器模块加入代码。而导出文件模块是负责将生成模块的代码写入一个 Vue 组件，提供给用户下载，此处的实现方案业界也有完整而安全的封装，故同样使用现有模块。

第 4 章　Web 前端组件智能生成系统的实现

在前一章中对 Web 前端组件智能生成系统设计方案进行了详细阐述后，严格按照设计方案来实现组件生成系统，在本章节对系统核心模块的进行实现，尤其重点介绍了拖拽绘制组件和生成组件，该系统的设计如下：使用人员首先打开网站进入系统，系统的主页即是组件拖拽页面。用户可以在左侧组件区选取需要的组件，并将其拖拽到中央的预览区，预览区中的组件也可以进行组件的删除修改。在拖拽完毕之后，可以对不同的组件进行属性修改，使其符合自己的要求。当准备就绪后，系统会判断系统是否已经有可以渲染的组件，点击进行生成。生成的组件可以进行 Vue 文件导出。

整体的系统时序图如图 4.1 所示。

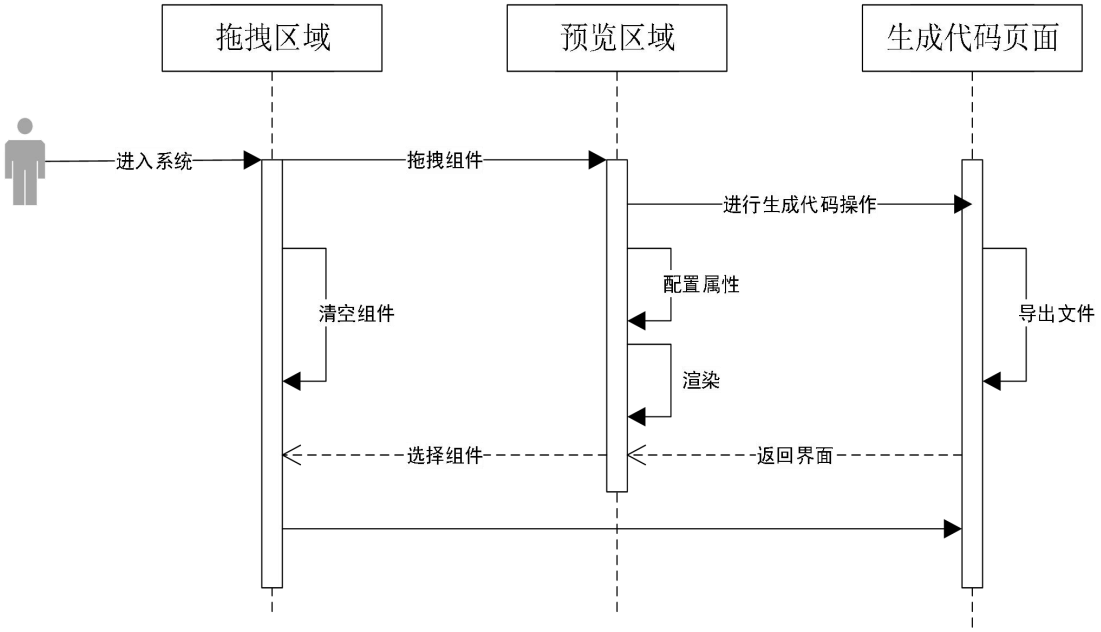


图 4.1 系统时序图

4.1 系统开发环境

实验代码采用处理纯前端语言编写，使用 Vscode 集成开发环境，运用 npm 进行开发工具包的统一管理，搭建可靠的 Web 前端组件智能生成系统。

系统开发环境的相关版本如表 4.1 所示。

表 4.1 系统开发环境表

基本配置	配置详情
操作系统	Microsoft Windows 10 专业版 64 位操作系统
集成开发环境	Vscode
开发语言	JavaScript
相关开发模块版本	Npm6.14.8

4.2 拖拽绘制模块

拖拽是本系统中最基础的部分，该模块负责拖动组件至预览区并组织组件的排列。拖拽绘制部分主要分为拖拽模块，属性模块，渲染模块 3 个模块。

4.2.1 拖拽模块

拖拽模块中主要使用了原生组件的一些属性和方法，draggable 设为 true 时，会将该组件设为可拖拽，dataTransfer 中存放了关键的数据如组件 type、组件位置、组件属性等等。为了让类之间耦合性降低，使用继承的方式实现该类。在 Component 父类中，只存放了 componentInfo 的组件配置属性，以及简单的打印方法 display()，而在子类 DropComponent 中，在构造函数里会将 componentInfo 存入 dataTransfer，这样做的原因是拖拽时触发的 drag 事件会读取到 dataTransfer，无法识别原先的 componentInofo。除此以外，只需配置 draggable 以及对应的拖拽事件、鼠标移动事件即可。该组件的类图如图 4.2 所示。

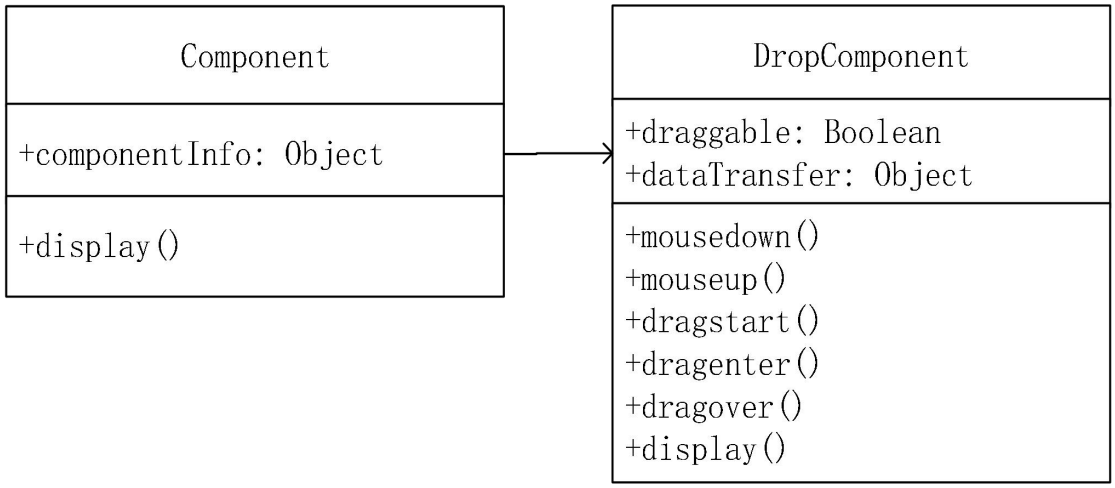


图 4.2 拖拽模块类图

该模块首先会遍历所有组件库的组件，查询是否还有没设为拖拽的组件。拖拽组件最基本的要求即是将组件的属性设定为 draggable，所以该模块的第一步就是把所有的 draggable 设为 true。设定完之后组件即可进行拖拽。但此时的拖拽只是单纯的拖拽，并不能使其松手后发生位置的改变以及触发其它事件，也不能检测到组件的数据，需要在进行额外的一些函数与数据设定[13]。当用户开始拖拽时，会触发 dragstart 事件，并且对于所有拖拽的事件中，都可以找到一个 dataTransfer 属性，拖拽数据拖拽的组件数据会被存入其中[14]。组件中有的是需要传递数据的，如输入的文本、标签 label 等，这些数据需要将他们都存放入组件的 dataTransfer 中，这样拖拽模块的基础准备工作就完成了（需要对所有组件都进行这些操作。流程如图 4.3 所示。

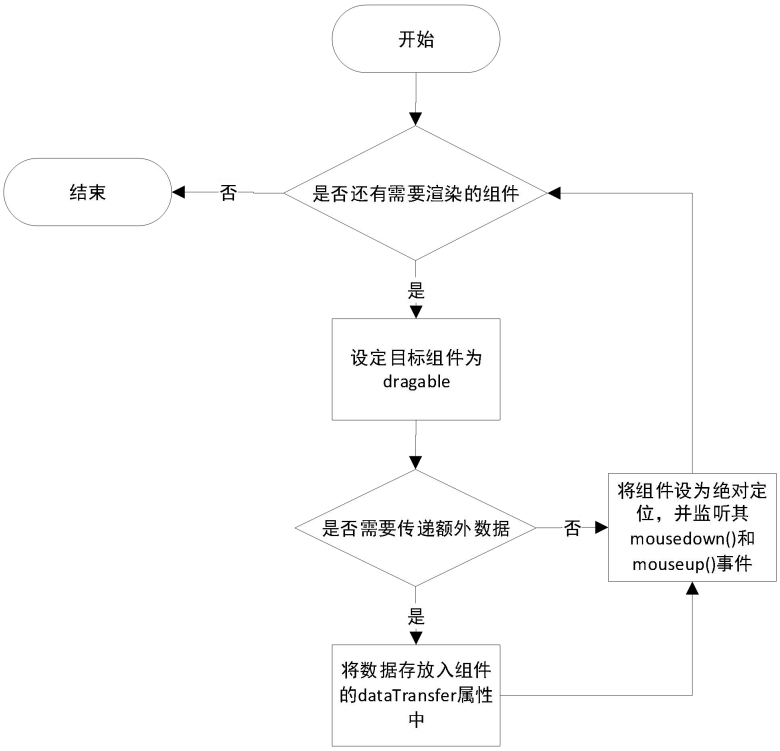


图 4.3 拖拽事件与属性

而对于拖拽放置区，也就是预览区，也需要进行相应的配置，dragenter 或 dragover 事件的监听程序用于表示有效的放置目标，也就是被拖拽项目可能放置的地方。在网页的大多数区域里，几乎都并不能进行放置。因为这些事件的默认处理时就被设定为了不允许放置。此时很容易想到 preventDefault()方法，通过它取消掉默认函数。具体的使用方法就是在预览区模块中，对于 dragenter 和 dragover 需调用事件的 preventDefault()方法来实现该区域可以进行放置的目的。

此时对该组件进行拖拽，即可检验到组件数据。有数据之后，要考虑组件的拖拽位置，这里使用 mousedown()和 mouseup()事件，当拖拽组件时，使用 mousedown 事件记录下当前的位置点 1，而当松开组件时，使用 mouseup 记录当前组件所在位置点 2，有了点 1 点 2 很容易计算出偏移的位置。最后将组件通过绝对定位，使用 top 和 left 调整位置，达到和拖拽时一致，即可完成拖拽操作，组件即可顺利停留在刚才拖拽的位置。过程如图 4.4 所示。

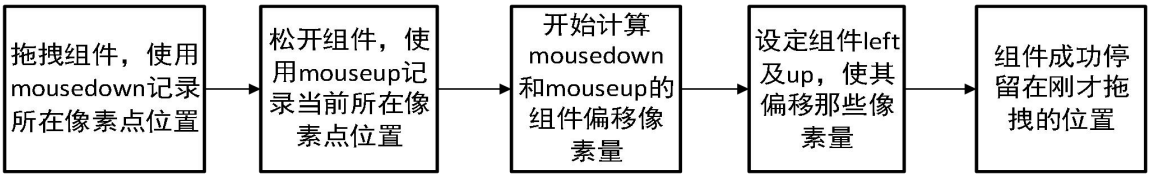


图 4.4 拖拽过程

4.2.2 属性模块

预览区域的组件点击后，即可在右侧的属性区域配置该模块的各项基本属性，

不同的组件可配置的属性不一样。当组件被拖入预览区时，开发者在属性配置模块输入想要配置的属性，然后当鼠标从输入框中离开并失焦时。数据将利用双向绑定直接同步给当前预览区的组件。这里的难点是双向绑定，Vue 中已有了现成的双向绑定实现，双向绑定的实现分为四部分。

1、实现一个监听者 Observer，它能够进行监听数据对象的所有属性，当数据发生改变时，会将最新值记录下来发送给订阅者。这里可以使用 Object.defineproperty()来实现，该方法可以具体配置一个对象，在其中可以设定 get 和 set 函数，来达到数据监听器 Observer 的监听效果。

2、实现一个解析器 Compile，这部分的目的是将模板中特殊的符号，替换成更新的最新数据。解析器需要解析对每个元素节点，再根据正则表达式进行替换数据。这里只处理最简单的情况，只对带有 '{{变量}}' 这种形式的指令进行处理，由于仅有 '{' 、 '}' 两种符号，解析器会获取到所有的 DOM 元素，凡是含有指令的节点，都会进行遍历，直接使用正则表达式 replace()即可对节点上的特殊符号进行替换[15]。

3、实现一个观察者 Watcher，他在 Observer 与 Compile 起了连接的作用，监听者会将所有的属性变化返还给观察者，让其收到属性变动通知，并执行对应函数，从而发生视图的更新。需要注意的是，订阅器 Dep 中必须需要观察者进行订阅，否则会收不到消息，此处的实现原理仍是使用了 Object.defineProperty()进行数据监听。这样一个简单的 Watcher 就实现了。

4、MVVM 入口函数，也就是整合以上三者而成。双向绑定的实现过程如图 4.5 所示。

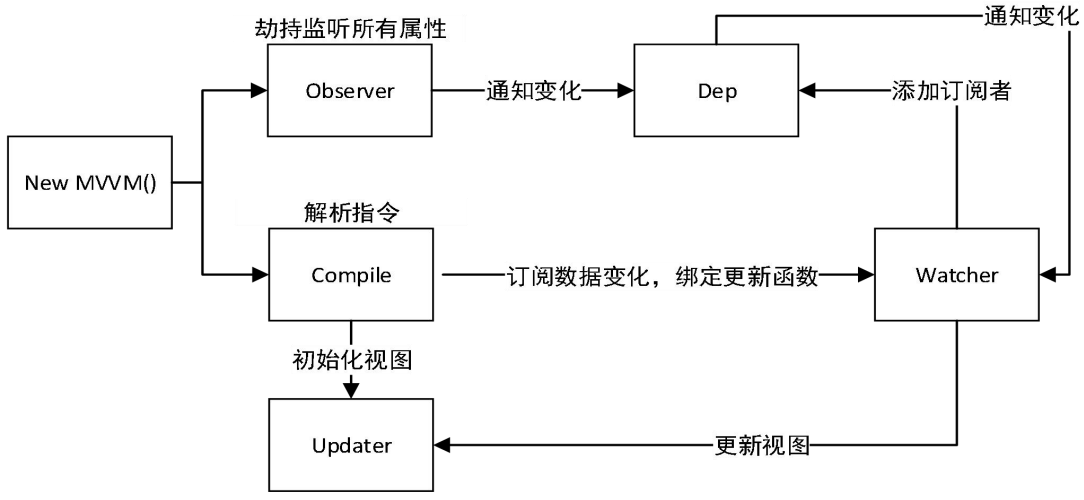


图 4.5 双向绑定过程

4.2.3 绘制模块

这是实现拖拽绘制的最后一步，在上一步将组件拖拽进预览区，并对其配置

属性后，通过部分设定好的参数校验保证属性数据的可行性，首先得到每个组件的拖拽位置，计算是否拖拽进入了预览区，若已在预览区内，则要计算其与其他组件的位置关系。若是一切正常，那么将会使用空白的模板组件，置于其中。接着需要将之前录入的属性数据进行传入，二次渲染组件，将完整的组件渲染出来[16]。

这里还需要说明，绘制时会涉及到一些先后顺序问题，因为是有顺序的将组件拖拽到画布中，所以分配图层层级时也会按照数据顺序来。这里举一个例子，当画布新增了三个组件 abc 时，那就可以推理得出它们在画布中的顺序为 [a, b, c]，它们的 z-index 就是 012，越后加的层级越高。

明白这里之后，就会明白如何改变图层层级了，只需要调整 componentData 数组中的组件顺序。[a, b, c] 三个组件为 abc，如果要下移 b 组件，调整数据顺序即可。如图 4.6 所示。

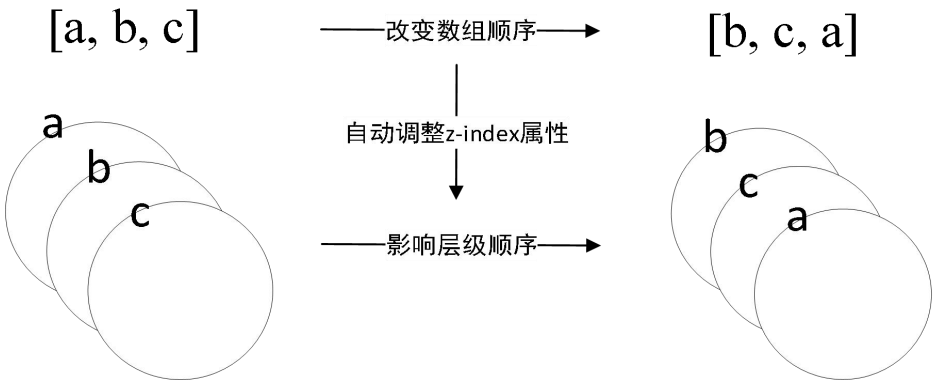


图 4.6 改变层级示意图

4.3 智能生成模块

在上一节得到了组件的具体位置和其类别后，将属性数据也写入了其中，此时只需要将该文件以代码的形式反馈给开发者，该系统的任务就结束了，在本模块实现了代码生成模块，编译器模块以及文件导出模块，这里对智能生成模块的实现过程进行阐述。

代码生成模块是 Web 前端组件智能生成系统的重要功能。开发者在使用系统时，最重要的目的便是将可视化系统转化为实际的组件代码，这就需要代码生成模块来实现。生成模块主要由模板文件实现[17]。对于每个不同的组件，会有不同的模板，当生成代码时，会将所有的组件属性导入模板中，再将其串起来整体进行拼凑，最终生成完整的代码。流程图如图 4.7 所示。

开始时，会进行 for 循环一个一个读取预览区的组件信息。预览区的组件信息最核心的是组件类型 type，读取到 type 时，会将该 type 跟模板文件中的各个组件模板进行对照。找到对应该 type 的模板文件时，便将该段模板拿出来，再获取组件中各属性的信息，将这些属性填入模板，而没有填属性的地方，将会自动填充默认值或空值。当所有组件都生成完毕时，会将其连接成一个长字符串，

还需要对最外层进行一次包装[18]。该系统使用了两种常用包装：页面和弹窗。根据选择包装的不同，会在字符串前后加入不同的代码段，达到实际效果。此时的字符串即是完整的组件了，再通过导出文件模块，将这些字符串都写入一个 Vue 文件中，并根据用户设定的文件名来生成一个完整的 Vue 文件[19]。桌面客户端展示图如下图 4.7 所示:

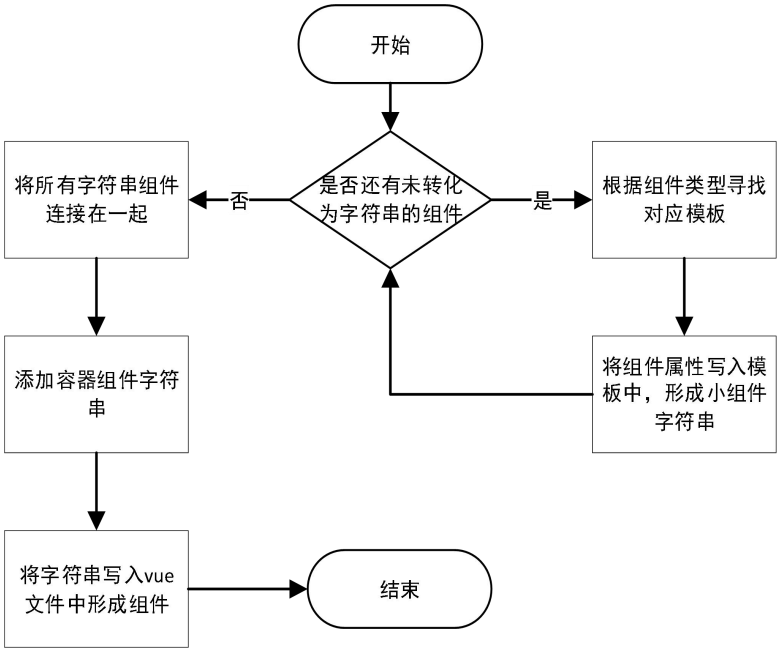


图 4.7 桌面客户端展示图

当有了生成好的代码后，就需要使用到编译器模块和文件导出模块，这两个模块的开发成本相对较高，权衡以后，使用了目前市场较为受欢迎的两个 npm 模块实现该功能。编辑器保证了将代码以较为美观的形式展示给用户，让用户可以自由选取和编辑，考虑到系统安全性，此时的编辑代码是无法进行保存和运行的。而文件导出相对简单一些，将所有代码处理为字符串，依次录入到 Vue 文件中，征询文件名后，会将该文件下载到用户的电脑上。

4.4 界面展示

4.4.1 拖拽绘制页面

系统的使用人员在进入系统后，在系统的左侧可以选择要使用的模块，在不进行后面的操作时，通过中心渲染区域可以观察生成的组件效果，人工进行组件渲染效果的观测。当需要使用系统的配置属性功能时，可以在加载组件后，点击右侧的属性面板，进行属性的配置，这时中心大屏中央渲染区会实时展示配置好属性的组件，同时如果对组件不满意时，可以使用组件右上角的删除按钮对其进行删除，也可以点击组件来重新配置组件属性。界面的渲染区上方有一个运行按钮，如果完成了配置，那么可以直接跳转到生成页面生成代码。具体页面如图 4.8 所示。

图 4.8 拖拽绘制页面展示图

4.4.2 代码生成页面

当在拖拽绘制页面选择配置组件完成时，即可点击工具栏中的运行，此时就会跳转到代码的生成页面。页面左侧为代码编辑器，可以在这里查看生成的组件代码，包括 template 文件、css 文件和 js 文件，右侧的区域则为组件实际渲染效果图，在这里可以看到组件实际渲染出来的样子，以防止渲染效果与自己预想效果不同，到下载文件再运行时才发现问题。除此以外，右上角还会有一个工具栏，其中会有导出和返回按钮。如果点击了导出，则会让用户填写文件名，之后将 Vue 文件下载到用户电脑上。若是点击返回，则会返回到上一个拖拽绘制页面。具体页面如图 4.9 所示。

图 4.9 拖拽绘制页面展示图

这就是系统页面的实现说明，到此为止对所有模块整个系统实现就结束了。

第 5 章　实验结果与分析

5.1 实验环境

该系统测试环境采用 Microsoft Windows 10 系统，CPU 为 i5-7300k 的四核处理器，系统主频为 4GHz，显卡为 GTX 1080ti，内存为 8GB，表 5.4 为系统实验环境的具体配置表。

表 5.1 实验测试环境

基本配置	配置详情
电脑型号	惠普暗夜精灵
运行系统	Microsoft Windows 10 专业版 64 位操作系统
CPU	Inter(R) Core(TM) i-7300HQ CPU @2.30GHz
运行内存	8.00GB
硬盘	512G

5.2 拖拽绘制模块测试

拖拽绘制模块功能的测试记录如表所示，通过控制环境变量，分别在不同顺序，不同输入值，不同拖拽顺序等环境条件下对拖拽模块的稳定性进行了测试。通过细致全面的分析后发现拖拽模块在各种条件下都较为稳定，与需求分析中的稳定性期望一致，符合设计要求。以下是过程中的各个测试用例。

拖拽的测试如下，首先正常进入系统，尝试拖拽左侧组件区域的组件，将其拖拽至右侧的预览区。若渲染顺利，那么预览区中将顺利生成该类型的组件，并且点击组件时，右侧会弹出对应的属性面板，工具栏会出现生成按钮。

拖拽组件测试用例表如表 5.2 所示。

表 5.2 组件拖拽模块测试用例表

条目	说明
用例名称	拖拽组件模块测试
用例目的	拖拽组件模块测试
预置条件	打开首页
测试过程	第一步：选中左侧组件面板的组件 第二步，拖拽至预览区
测试结果	组件会生成到预览区，并出现属性面板和生成按钮

按照流程操作后，组件成功生成至了预览区，且配置属性、生成按钮均正常出现。此时进行对属性的配置可以正常实时响应到预览区中，对组件拖拽并重排顺序也未出现问题。当删除和新增组件时，组件正常消失出现。

测试结果符合预期，渲染结果图如图 5.1 所示。

图 5.1 渲染结果图

5.3 智能生成模块测试

智能生模块功能的测试记录如表所示，通过控制环境变量，分别在不同顺序，不同输入值，不同拖拽顺序等环境条件下对智能生成模块的稳定性进行了测试。通过细致全面的分析后发现智能生成模块在各种条件下都较为稳定，与需求分析中的稳定性期望一致，符合设计要求。以下是过程中的各个测试用例。

代码生成模块测试流程如下，在预览区已有组件的情况下，直接点击生成，分别选择生成弹窗和生成页面，此时会跳转到生成代码界面，页面的左侧为代码编辑器，里面是生成的代码，该编辑器可实际编辑代码，但不可运行代码。而右侧则是此段代码的效果图。效果图的上方分别有导出 Vue 文件和返回两个按钮。点击导出后，会弹出请命名该文件，当填写好文件名后，点击确定。此时组件会转化为 Vue 文件，下载到电脑中。如果点击返回，则保持刚才的组件位置，恢复到之前的拖拽绘制界面。代码生成模块测试用例表如表 5.3 所示。

表 5.3 代码生成模块测试用例表

条目	说明
用例名称	代码生成模块测试
用例目的	代码生成测试
预置条件	预览区已有组件
测试过程	预览区已有组件的情况下，点击生成
测试结果	跳转至生成代码界面，左侧为代码，右侧为渲染图

按照流程进行操作后，成功生成了代码，左侧的编辑栏正常显示，可编辑和复制代码，右侧的预览组件也正常展示。当点击导出并命名后，该文件正常下载到了电脑中。点击返回时，成功退到了上一步的拖拽绘制界面，此时进行其他操作均为出现问题。

测试结果符合预期，生成结果图如图 5.2 所示。

图 5.2 生成结果图

5.4 测试结果与分析

本次测试主要对两个模块进行了验证，分别是拖拽绘制模块和智能生成模块。拖拽绘制模块是智能生成系统的基础，在拖拽组件的地方展开了大量测试。当用户正常进入系统后，左侧组件区域的组件的拖拽是测试要点一，将其拖拽至各种地方，均可以正常拖拽并且不影响后续流程。拖拽顺利完成后，预览区中将顺利生成该类型的组件便是测试要点二，据测试来看，对不同的组件确实都正常的进行了渲染，测试要点三则是属性面板。点击组件时，右侧会弹出对应的属性面板，根据正常流程，调整不同的属性，组件都出现了对应的变化，测试结果符合预期。

在各种拖拽组件的情况下，均未出现明显问题。点击生成后，就会会跳转到生成代码界面，页面的左侧为代码编辑器，里面是生成的代码，这是最基本的框架。该编辑器可实际编辑代码，但不可运行代码。而右侧则是此段代码的效果图。效果图是测试要点一，效果图的上方分别有导出 Vue 文件和返回两个按钮。点击导出后，会弹出请命名该文件，当填写好文件名后，点击确定。此时组件会转化为 Vue 文件，下载到电脑中，下载的文件则是测试要点二，经过几次下载测试，都获得了较为合理的组件文件，测试结果符合预期。

本次实验详细的测试了拖拽绘制模块和智能生成模块两个模块，针对不同情况、不同组合，该系统都展现出了较为合理的结果。证明了该系统有较强的健壮性和稳定性，对于正常的使用来说完全满足。

第 6 章　总结与展望

6.1 工作总结

本文是依据软件开发流程：系统需求分析、系统总体设计、系统详细设计和系统测试四步完成。本论文所做的工作主要有：

在系统需求分析工作中对开发进行分析，应该明确系统必须实现什么功能这个问题，对系统提出完整准确的要求。系统需求分析主要介绍功能需求分析和非功能需求分析，通过流程图，将每个环节以图表的形式进行展示。在系统需求分析后，以需求为基础对系统进行总体设计。系统的总体设计包括系统总体框架设计和数据库设计，在本章中，大体规划了整体布局与设计思路。系统详细设计和实现主要是对系统的功能模块进行详细设计并编程实现。本文的核心功能是前端组件智能生成，本文主要采用的 Vue.js 的框架实现，将组件区域布置于左侧，预览区域布置于中间，绘制完成以后，能很明显的观察到绘制结果。最后系统经过在各种条件下，对各模块的功能进行了详细完备的测试，达到预期效果，符合之前预期的构想。

6.2 展望

由于软件开发能力和毕业设计的时间有限，本文实现的 Web 前端组件智能生成系统还有很多缺陷和不足，希望在今后的学习过程能够进行分析解决。通过这次毕设工作，我的代码编程能力得到提高，并且认识到自己能力的不足，我会在今后的研究生学习中努力学习，多动手实践，不断提高自己的专业能力。 在本文的基础上，可以进一步的研究的工作有：

对拖拽进行优化，提升使用者的操作体验。经过实验发现，目前系统对于拖拽时的各个像素点位需要进行大量计算，可以优化此间算法，来提升拖拽的流畅度。完善组件的导出功能，目前对于导出文件，仅仅停留在导出 Vue 文件，未来可能会增加导出其他类型文件的功能，甚至兼容其他前端框架（如 React）的文件。可以将该系统的输入值进行一定的识别判断，对于每个 input 框都进行正则校验，这能显著提高系统稳定性。

参考文献

[1]Jörg Krause. Developing Web Components with TypeScript[M].:2021-03-18.

[2]胡如乐,张亮,张倩,司福利,肖睿.基于微服务基础组件的前端开发技术应用研究[J].电子测试,2020(20):84-86.

[3]杨仁宝,杜兵,杨操,焦蓬斐,周淦.基于组件技术的通用指挥系统框架设计[J].信息技术与网络安全,2020,39(09):79-82.

[4]窦健. 基于组件化电力监测管理系统研究与实现[D].西安科技大学,2020.

[5]韩思雨. 组件化可配置 B2B2C 平台供应商前端系统的设计与实现[D].北京邮电大学,2020.

[6]王东昊. LVC 资源组件生成工具开发[D].哈尔滨工业大学,2020.

[7]战仕霞. 基于 React 的信息管理系统前端自动化构建工具的设计与实现[D].北京交通大学,2020.

[8]刘甜. 基于 Web 前端组件化的消防维保管理系统的设计与实现[D].西安电子科技大学,2020.

[9]陈子豪. 设计图转换组件化 HTML 代码生成系统的研究与实现[D].北京工业大学,2020.

[10]宋雅. 基于 Web 的大屏数据可视化系统的研究与实现[D].北京邮电大学,2020.

[11]李成仁.基于 Vue.js 的单页面 WebGIS 可视化框架研究与实现[J].地理空间信息,2020,18(05):83-86+98+7.

[12]孙鸿江. 企业级应用中的性能提升和可扩展架构研究[D].长江大学,2020.

[13]王叶加. 基于 JavaScript 的 React 一站式智能开发工具的设计与实现[D].北京邮电大学,2020.

[14]张根,蔡永香,高静文.基于 React 组件快速构建网站前端[J].电脑知识与技术,2019,15(15):119-121.

[15]杜艳美. 基于 web 前端的性能优化框架模型研究[D].西南科技大学,2018.

[16]王萌,田杨,李宁宁.组件化 WEB 前端架构设计与实现[J].电脑知识与技术,2018,14(30):77-79.

[17]Alinaghi Marzoughi,Rolando Burgos,Dushan Boroyevich. Optimum Design Guidelines for the Modular Multilevel Converter in Active Front-End Applications: Considerations for Passive Component Reduction[J]. IEEE Power Electronics Magazine,2018,5(2).

[18]王志任. 基于 Vue.js 的开发平台的设计与实现[D].广东工业大学,2018.

[19]何奔. 基于 Web Components 的跨终端商城系统的设计与实现[D].北京邮电大学,2018.

致谢

四年一晃而过，我的大学即将结束。大学四年，在学校和实验室收益良多，让本是惫懒的我也更进一步有了各方面提升，路漫漫其修远兮，吾将上下而求索。在未来的日子里，希望我能更上一层楼。在此对我的导师李松江、审查老师徐春凤、各位帮助我的实验室老师以及陪伴我的各位朋友们致谢。