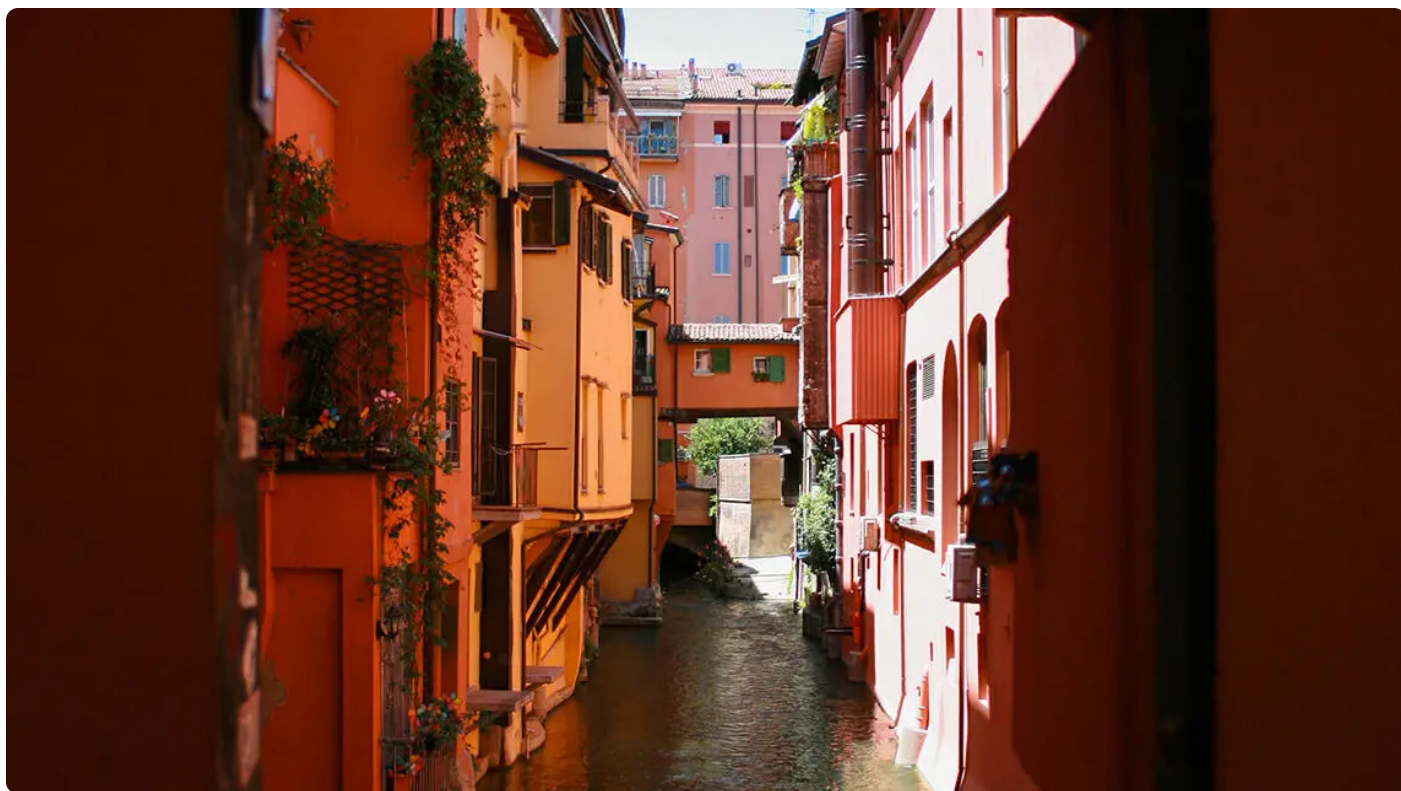


加餐 | 实战篇思考题答疑（下）

2023-02-22 杨文坚 来自北京

《Vue 3 企业级项目实战课》

[课程介绍 >](#)



讲述：杨文坚

时长 01:20 大小 1.21M



你好，我是杨文坚。

实战篇我们就学完了，在这个实战阶段中，我们围绕着搭建页面的核心业务逻辑，页面搭建、页面编译和运行、页面发布流程、页面版本管理、页面渲染方式这五大模块，进行了功能设计和技术实现。最后，我们也延伸学习了前台页面性能优化和日志监控，为增强篇做提前的技术探路。

这五个模块功能的技术实现，基本囊括了我们在基础篇和进阶篇学过的知识点，比如 Vue.js 组件的多种模块化方案、Vue.js 的非编译渲染方式等等，同时，我们也分析了很多 Vue.js 体系之外的技术方案，比如用 JSON Schema 验证 JSON 数据、用 ESTree 来动态生产 JavaScript 代码等等。

通过串联这五个模块技术实现和功能链路，一个完整的运营搭建功能链路就此诞生。实战篇的成果，也为我们接下来的增强篇提供了实际案例和材料。

今天我们对实战篇后半部分统一答疑，如果对课程中的知识点还有其他疑问，也欢迎你留言。

24

课程：🔗 [24 设计运营搭建页面的数据结构](#)

提问：既然可以把 TypeScript 数据类型编译转成 JSON Schema，那么有没有技术可以把 JSON Schema 编译转回 TypeScript 代码数据类型呢？

答案是有的，我们通过 `typescript-json-schema` 这个 npm 模块，把 TypeScript 的数数据类型声明，转成了 JSON Schema。技术社区里也有开发者实现了 JSON Schema 编译 TypeScript 数据类型的 npm 模块，例如这个 🔗 <https://www.npmjs.com/package/json-schema-to-typescript> npm 模块，可以把 JSON Schema 转成 Typescript。

扩展介绍一下 JSON Schema 和 TypeScript 的互相编译转换原理，两者原理都是类似的。首先，都是将一种语言先编译成其语言的语法树（AST），这个语法树其实是一种 JSON 格式的数据、

然后解析这个语法树（AST）转成目标语言的语法树。最后，将目标语言的语法树编译成语言内容。

25

课程：🔗 [25 设计和实现 Vue.js 运营后台的搭建功能](#)

提问：在我们的页面搭建技术方案设计中，为什么渲染物料组件考虑用 AMD 模块格式，而不选用 ESM 模块格式呢？

代码案例中，我们选择了 AMD 模块格式，主要是考虑到共用同一个 Vue.js 运行时。因为在 AMD 运行时环境中，可以通过 `define()` 这个全局方法，将当前页面应用的 Vue.js 运行时对象注册到 AMD 的运行模块中，提供给动态加载的物料 AMD 格式组件使用。

那 ESM 模块格式的物料组件，就不能和页面搭建功能应用，共用同一个 Vue.js 运行时吗？

是可以的，但是实现起来会比 AMD 更加繁琐。因为 ESM 格式物料组件中，使用 Vue.js 运行时是通过 `import` 的方式来使用，这就意味着要在当前页面 HTML 上注册 `importmap`，来让

ESM 模块能通过 `importmap` 别名，加载到 `Vue.js` 运行时。

但是这个运行时要跟页面功能应用的 `Vue.js` 保持一致，最低成本方式就是要让运行时放在页面 `window` 全局变量上，这时候就要在编译配置上做处理。如果不想 `Vue.js` 挂载全局变量使用，那么整个页面在生产环境中所有代码，都需要 ESM 模块格式才能共用 `Vue.js` 运行时。

你可能会有疑问，**为什么物料动态渲染和页面应用，一定要共用同一个 `Vue.js` 运行时呢？**

因为如果不共用同一个 `Vue.js` 运行时，会带来一些 `Vue.js` 的渲染异常或者警告，甚至带来一些无法预料的问题。

所以我们用 AMD 格式的物料 `Vue.js` 组件，主要是考虑到“物料动态渲染”和“页面应用”，能低成本共用同一个 `Vue.js` 运行时。这也是前几节课为什么要保留多种模块格式的原因。

26

课程：🔗 [26 页面编译和运行，设计 `Vue.js` 搭建页面的渲染策略](#)

提问：页面渲染策略中，**Bundle** 文件渲染不能兼顾“用户体验”和“技术稳定性”吗？渲染策略必须做降级处理，牺牲用户体验，变成物料独立加载渲染吗？

Bundle 文件渲染，理论上，可以做到兼顾“用户体验”和“技术稳定性”，但是要在基于 **ESTree** 拼接入口代码中，做很多“容错”代码的工作。

比如，每个物料组件，在组装页面的时候，要独立做容错处理。又比如，在处理数据和渲染步骤中，要做一堆容错处理。这相当于，组装的页面要基于 **ESTree** 拼接很多容错代码，但这只是理论上可预测的容错处理，实际过程中，不一定能预测和覆盖到所有可能出错的场景。

与此同时，**Bundle** 文件渲染页面，其实就一个 **JavaScript** 来渲染页面。但是，我们都知道，**JavaScript** 执行环境是单线程的，一旦在执行过程中遇到异常错误，很大可能会导致 **JavaScript** 程序中断。

所以课程里的渲染策略，就是一旦在 **Bundle** 渲染遇到错误了，选择牺牲用户体验，改成多个物料组件独立加载其 **JavaScript** 文件进行独立渲染，隔离执行代码，避免错误干扰。

但这种策略**也不是绝对稳定和安全的，只能说是尽量保证功能的可用性**。这也是软件工程里经常提到的一个概念“没有银弹”，没有绝对完美的技术解决方案。技术工作中也是一样的场景，没有“绝对的最优解”，只有“相对的次优解”。

27

课程：🔗 [27 实现 Vue.js 搭建页面的发布流程](#)

提问：我们的流程设计是线性的，测试、预发、生产三个流程节点操作是串行。那么有没有并行多环境的操作，例如同时多个测试环境、多个预发环境的流程设计？

发布流程多套并行环境，是可以存在的，但是不一定所有流程节点都适合多套并行环境。

例如测试节点，可以考虑支持多套并行环境。因为是面向程序员进行开发测试的，可能一个项目同时有多个程序员进行开发代码，为了避免出现互相干扰，可以进行多套测试环境，并行使用。

但是，例如预发布节点，就不建议用多套并行环境。因为预发布节点已经是验收功能环节，临近上产线给客户使用，必须保证其唯一性，也就是保证大家验证功能的时候，都在同一个环境中验证，出问题可以快速定位。如果是多套并行预发布环境，没办法确定哪一个环境能代表模拟生产环境的标准，验收环节的权威和保证就大打折扣。如果一个并行的预发环境出问题，怎么确定生产环境，也有类似问题。

而且，大部分企业中，预发环境为了能正式还原生产环境，有时候还会使用真实生产数据，如果多套预发布环境，就会对真实生产数据带来干扰，甚至产生脏数据。

当然，事情也不是绝对的，如果能考虑周全，保证流程的健康稳定，发布流程的多套并行环境也能实现，只是你要对比实现的投入成本和最终收益，看看值不值得这么做而已。

28

课程：🔗 [28 前台页面版本化管理，实现搭建页面的迭代策略](#)

提问：运营搭建平台，除了用多版本回滚的措施外，还有其它的措施来快速对问题页面进行恢复吗？

答案是有的。

首先，我们回滚页面，主要是解决页面出问题不可用的问题，那么核心就是要将页面不可用变成可用。而且，这个恢复过程要尽量快速实现，避免出现客户的长时间不能使用。

有个两个参考方案。

- 第一种，利用之前的多种模块格式渲染方式。可以按照需要，一旦遇到用户反馈问题，就将页面从 **Bundle** 模块，切换到 **ESM** 或 **AMD** 的模块格式，通过模块格式文件的动态加载，隔离每个物料渲染，尽可能将错误隔离出去，保证大部分功能能正常使用。
- 第二种，给每个页面指定一个固定的稳定版本内容，一旦反馈出问题，就快速执行稳定版本内容进行覆盖，让用户能使用到页面基础功能。

29

课程：🔗 [29 设计前台页面的渲染策略](#)

提问：课程中，前台服务的浏览器端渲染和服务端渲染，都是基于 **Node.js** 服务的。假设随着业务发展，我们要对前后台服务架构进行分离，面向客户的前台服务要变成 **Java** 开发和维护，面向内部员工的后台保持用 **Node.js**。那么如何保证服务端渲染的功能，从 **Node.js** 环境平滑转换支持 **Java** 服务端的服务端渲染？

服务端渲染的本质是在服务端进行拼接 **HTML** 代码，再将 **HTML** 响应返回给前端浏览器。

拼接 **HTML** 代码核心是在“模板”中填充“数据”，编译成 **HTML**。所以**要让服务端逻辑从 Node.js 迁移到 Java，就是需要解决“数据”和“模板”的迁移问题。**

数据都是来源于数据库，**Java** 有很多工具可以转换数据库的数据，那么剩下的要素就是“模板”。**Vue.js** 的模板代码最终编译成 **JavaScript** 代码，我们可以基于这个 **JavaScript** 代码，利用 **ESTree** 来进行解析处理，转成 **Java** 生态里的模板框架代码，例如 **Velocity**、**FreeMarker** 等等。

模板转化的主要工作量，只需要写一个模板语法的转换器就好。

提问：前端性能优化，很多都是关于优化静态资源请求的控制，我们是通过人工写代码懒加载的方式来处理，业界还有其他优化思路或方案吗？

答案是有的，**让服务端来控制资源是否要合并和拆分加载**。你可以参考阿里提供一个 Nginx 层面的插件，方便服务端或者 CDN 来统一控制管理静态资源请求的合并或者拆分。文档在这里：🔗 https://tengine.taobao.org/document_cn/http_concat_cn.html。

具体方式就是在服务端控制是否要合并请求，例如有三个资源。

📄 复制代码

```
1 <link ref="stylesheet" href="https://example.com/style1.css" />
2 <link ref="stylesheet" href="https://example.com/style2.css" />
3 <link ref="stylesheet" href="https://example.com/style3.css" />
```

可以通过服务端使用这个插件处理，控制静态资源请求是否要合并。如果合并了，就变成一个静态请求操作。

📄 复制代码

```
1 <link ref="stylesheet" href="https://example.com/??style1.css,style2.css,style3"
```

31

课程：🔗 [31 实现页面实时监控与问题定位](#)

提问：我们学习了页面级别的跳转链路日志跟踪，那要实现页面某个模块曝光、点击或其他操作的行为跟踪定位，如何设计这个日志收集功能？

要实现页面某个模块曝光、点击或其他操作的行为跟踪定位。首先，需要标识和区别页面上所有模块。这里可以有个模块注册功能，对页面的模块进行注册记录标识。如果页面模块维度跟物料模块维度重叠，可以在搭建页面时候，利用模块的 **uuid** 作为模块的标识。

如果要上报模块“曝光”的日志，就要监听页面的全局点击和滚动事件，判断模块是否进入可视区间，如果进入可视区间，就等于“曝光”了，可以上报日志数据。


如果要记录模块“点击”的日志，就监听页面全局点击事件。如果页面某个模块被点击，就进行事件的冒泡和捕获，记录点击所在的模块，上报日志。

思考题就解答到这里。

在实战篇，我们已经完成了运营搭建平台的核心功能，但我们不能止步于完成功能，仍要保持技术追求，以更高效的技术方式，让项目工作更加轻松。

下一节课，我们开始增强篇的学习。

分享给需要的人，Ta购买本课程，你将得 18 元

 生成海报并分享

 赞 1  提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 31 | 前台页面的日志监控：如何进行页面实时监控与问题定位？

下一篇 32 | 单元测试：如何打造Vue.js和Node.js全栈项目的单元测试？

精选留言

 写留言

由作者筛选后的优质留言将会公开显示，欢迎踊跃留言。