# resume\_project

## 自我介绍

面试官你好，我叫唐亚婷，来自四川南充，我的技术栈主要是Vue，做过uniapp端，面向C端用户，B端用户，后台等相关的项目，

## **主题色切换功能** 3. 语言国际化功能 4. 关于Echart中问题 5. 虚拟列表技术

虚拟列表是一种技术，他只渲染可视区域内的列表项，而不是渲染整个列表，当用户滚动容器时，虚拟列表会根据滚动位置和可视区域的大小去计算出当前应该显示的列表项。

从后端请求获取到10000条数据，将它存在data中，而初始渲染list为data的前containerHeight / itemSIze条，通过ref去获取元素，并且获取到元素的scrollTop，假设每一条数据的高度已知，且容器的高度已知，通过scrollTop / itemSize 和 (scrollTop + containerHeight ) / itemSize 可以计算出可视区域内要渲染的元素索引。

对于高度不固定的情况下，可以设置一个默认的虚拟itemSIze，通过this.$refs.container.clientHeight可以获取容器高度，然后计算出一个虚拟的positions数组，这个数组包含了从data数据的每一项的height，index，top，bottom，然后在update钩子函数中去更新positions数组中当前可视区域内item的真实高度和位置，这个updateItem可以通过this.$refs.items来获取所有的nodes，并遍历node节点通过getBoundingClienrRect获取它的height，oldHeight是我们默认设置的高度, 当oldHeight和height不等时，去重置position里的height为height， bottom 为oldHeight - 差值。然后便宜量等于start位置的bottom值，更新之后再重置偏移量。

<https://juejin.cn/post/6844903982742110216#heading-4>

在从position 中查找start值 和 end 值的时候，用到了二分查找算法。

用到了算法：二分查找算法

Positions.some(item => item && item.bottom > this.scrollTop)

由于positions中的数据改为二分查询算法(O(logn), 最差为O(n)), 而Array.some的时间复杂度为O(n)

找第一个大于value的值:

function(position, value) {

let start = 0

let end = position.length;

while(start <= end) {

const centerIndex = Math.floor(start + (end - start) /2)

if(position[centerIndex].bottom == value) {

return centerIndex + 1

}

if (position[centerIndex].bottom < value) {

start = centerIndex + 1

} else {

end = centerIndex

}

}

return start

}

找到第一个小于value的值:

function(position, value) {

let start = 0

let end = position.length

while(start <= end) {

const centerIndex = Math.floor(start + (end - start) /2)

if(position[centerIndex].bottom == value) {

return centerIndex + 1

}

if (position[centerIndex].bottom < value) {

start = centerIndex

} else {

end = centerIndex - 1

}

return end

}

}

追问：一个列表项元素很大时，出现跳跃式滚动怎么优化，白屏怎么优化.

白屏的情况可能是由于数据请求延迟导致的，这是时候我们可以设置一个虚拟数据放置在页面上，等请求完成再去移除虚拟数据，更新为真实数据。还可以采用增加加载提示。除此之外，还可以应用预加载技术，提前加载当前页前一页和后一页的数据。  
**5. 首屏优化(包体积，首页，白屏)**

<https://juejin.cn/post/7188894691356573754?searchId=2023121210475077B0E9C9480B03FA32F9>  
冗余资源分析，bundle分析，删除掉一些重复冗余的资源以及对静态资源优化，第三方资源放在public下，自己的资源放在static下方便webpack进行处理。对于图片压缩，开启gzip压缩，svg压缩，针对图片，还可以处理图片格式，使用webp格式图片在同等体积下提高性能，针对组件库按需引入，很多时候我们为了方便，使用组件的时候直接一次性全局引入，但是实际上所使用到的组件并不多，导致很多未使用的组件库被打包，开启treeShaking去除掉多余的代码，除此之外，对于分包策略进行优化，拆分依赖，避免文件过大拖慢页面展示，对路由懒加载，组件懒加载，利用CDN进行网络加速，还可以在感官方面，对于首屏增加骨架屏，路由跳转的时候增加加载动画等，图片懒加载等，还有预请求(rel = prefetch)，预加载(rel = preload)来优化用户体验，提前获取和缓存资源。

**6. 复杂表格联动，多级选择框数据联动，动态配置数据项，表格内数据操作。**

处理数据用到的算法

1. 树型结构转换

组件封装：考虑配置，考虑权限，考虑安全，考虑复用。

1. <https://juejin.cn/post/7300872843587469327?searchId=2023121214192189C1B7804417AF0A5345#heading-7>

考虑配置项，哪些属性，方法，事件，方法

为何要进行组件封装：

1. 二次封装是为了样式统一
2. 方便复用，降低耦合性，使单个组件和功能的变更不会影响其他部分
3. 增强安全性，可以隐藏数据和实现的细节，防止外部的非法访问和修改
4. 方便维护，将修改局限在组件内部。

组件拆分和组件封装的区别：

a: https://segmentfault.com/a/1190000039846269#item-5

1. 组件拆分更多关注于如何将大的功能模块划分为小的，可管理的单元，以便管理和复用。
2. 组件封装更多则关注于如何将组件的内部细节隐藏起来，提供清晰的接口，以保证组件的独立性和安全性。

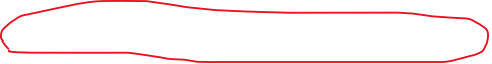
封装流程组件：

我们的业务是需要在11个流程页面里面都展示这个流程组件，并且每个流程页面都可以推动流程的更新且数据的展示来源于同一个接口，内容是根据流程阶段作区分的，所以就封装了一个流程组件，在里面展示流程执行阶段图，并且根据流程阶段调接口更新该流程数据传递给父组件，父组件可以直接渲染数据，除此之外在里面监听了流程阶段更新事件，当用户操作流程进入到下一步，触发流程更新，流程组件则会自动执行重新获取新流程信息，以导致路由更新。

拆分公共的人员选择弹窗组件：

滑动选择组件：

动态表，表头的封装(涉及到动态表头的中英文翻译) （明天尝试实现一下）



针对动态表格进行配置主要是涉及到业务中需要，表格头根据不同的目标项进行配置，然后涉及到动态表头的渲染以及动态表头下的表格项动态框的操作，还有个性化的操作项。我只需v-model:taskInfo传入当前这条目标项任务数据，当保存的时候触发update:taskInfo更新。

**7. 处理权限管理数据的时候**

后端返回的JSON路由菜单结构，需要处理成树状结构，用到了JSON转树的算法。

// 周三来不及来，周四，周五两天搞！！！  
**8. 项目搭建中的问题**

1. **水印组件封装**
2. **uni.request的封装**
3. **ChatGPT接入问题**
4. **单点登陆**
5. **generator格式接收**

**不着急，周六，周日再搞直播的项目！！！**

**周一，周二，周三复习手写+八股+算法！！！**