3月20日第一天

初始化项目, 安装, 资源, 预备

学习stylus 配置stylus和stylus-loader

学习图标字体转化配置 svg iconMoon

配置网络 安装 axios 和mockjs 简单粗暴的在mockjs导入本地data.json文件

了解移动端物理像素和逻辑像素 学习在手机上查看开发的网页

问题:

各种文件应该导入到什么地方以及别名的使用

公共文件在main.js中导入, 有多个同类文件如styl文件需要导入时,先用一个index文件全部装起再导入

3月22日第二天

Sticky footers 布局

min-height

**水平居中**

行内元素 text align

块级元素 margin 0 auto

**垂直对齐**

vertical-align 适用于inline-type元素, 默认baseline

图片初始baseline 需要有文字才有（font size不为0即可）具体高度应该是取决于与最大的font size

注 父元素为0 子元素中有不为0到还是会有baseline

\*\*行内元素就是内容text 可以通过text-align水平对齐; 通过设line-height==height垂直对齐

但是图片和文字相遇会遇到问题, 图标字体会遇到问题, 中文和英文可能也有问题(待验证?)

Solution: 顶部对齐

设置父元素font-size为0 设置子元素 vertical-align为top

居中对齐, 也就是图片和文字的中线对齐

关键在于行高, 比如将行高和图片高度设为一样的

\*\*\*文字在行盒中是等分的（字体大小超过行高也是如此）, 行内元素高度取决于行高

\*\*\*文本高度既不等于行高也不等于font-size(图标字体除外==font-size)

感觉图片是baseline 而文本则与图片的vertical align设置有关 设置为baseline时在下方 top手机在上方 middle在中间

span标签有无文字位置会变化

总结: 以上问题花了比较多时间 情绪问题是主因 不相信自己 不相信自己所做的是对的是值得 所以没法行动 所以不去行动 那么,不去行动 究竟如何来验证错对? 况且, 为什么觉得自己做的不对呢, 为什么呢. 其次是恐惧 觉得这个内容很困难 没法掌握 学不会 这种不可知感引发了恐慌 但其实不是这样 在接触这个知识一天后就觉得它不难了 然后一段时间的学习分析 已经算是了解的很透彻了 它就是一个普通的知识 是常规操作 没什么特别的 为什么开始害怕它呢, 可能是因为很早以前研究baseline没有搞明白 留下的后遗症 所以, 要提醒自己不要钻牛角尖 非要一时间把一个东西搞的明明白白, 结果一根筋得不到结果 影响后面的学习 要循序渐进 随着知识的积累 时间的推移 一点点弄透彻

学习知识的阶段

为什么是这样 怎么能这样的

好像是这样

好像不是这样

确实是这样

怎么可能不是这样

计算机领域中 任何知识都不是特别的,不是特例, 都是有理可循的, 都是可以理解, 可以掌握的 要说哪里会有困难 知识的应用 知识间的联动与冲突 但是要相信 基础扎实 经常实践 就能解决绝大部分 还有一少部分别人肯定遇到过解决过 任何一个高发的痛点问题都可以通过百度获得答案 不行还有谷歌

学点东西而已 又不是要创造东西 能有多难 怕什么

所有的知识都会从新鲜的突出的变成我知识体系中普普通通的一片

3月24日第三天

如果确认console报错的部分没有问题, 那么可以重新编译,刷新, 解决别的报错, 获取能解决问题. 没有什么是一次重启解决不了的, 如果有, 就两次

组件命名问题: header detail 会报错, 最好用两个单词, 或者前面加v

不能在data中引用计算属性, 应该是不能将对自己有依赖的计算属性赋给自身, 要不然就无限死循环了(这一点待验证)

3月25日第四天

Vue router

Vue路由通过两种方式应用

<router-link to=”/path”><router-link>

<router-view></router-view>

Click事件 ---> this.$router.push(/path)

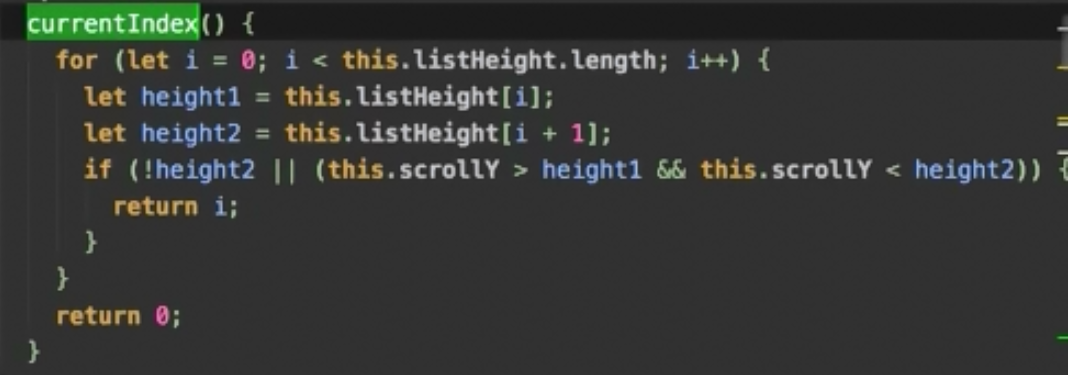
<router-view></router-view>

router重定向和别名

<https://blog.csdn.net/AprilCC/article/details/78500104?depth_1-utm_source=distribute.pc_relevant.none-task&utm_source=distribute.pc_relevant.none-task>

问题待解决: 使用箭头前进后退 tab标签不切换





学习了! 标签滚动联动 数组前后两项比较确定落点

3月26日第五天

事件总线bus：

//vue原型链挂载总线

Vue.prototype.$bus = new Vue();

this.$bus.$emit("vaPage",value); //很像父子之间传值的方法

this.$bus.$on("vaPage", v =>{ //vaPage传的时候的key是什么接收就必须是什么 console.log(v); //v是传来的值，可以接收多个参数 })

使用$bus的时候在接受bus的组件中别忘了再beforDestroy函数中销毁bus，不销毁的话会一直叠加的调用这个方法：

beforDestroy（）{

this.$bus.$off("vaPage"); //当这个组件销毁的时候bus也跟着一起销毁

**计算属性** 计算属性中的getter方法会在属性使用时(也就是用在dom中, 可以是dom的任何地方, 值 属性 甚至事件函数的实参)和使用时其所依赖的值发生变化时调用, 即使依赖值的变化没有改变属性的值, 也会调用并触发数据更新

\*\*\*计算属性的初次渲染也会触发updated钩子

左右菜单联动效果

第一步 在goods组件中

用div分别包裹两个菜单组件;

安装并导入better-scroll 为BScroll;

在methods中定义函数initScroll:

1, 通过foodScroll = new BScroll(el, {options})新建better-scroll对象, 并将这个对象传到left-menu中;

\*\*\*元素通过$refs.ref获取

2, 设置对foodScroll.on(“scroll”, (pos)=>{})设置对滚动事件的监听(options中需设置probeType: 3), 并将y轴上的滚动距离传给pos.y传给left-menu组件;

3, 在created钩子中, 通过this.$nextTick(()=>{initScroll()}), 在dom渲染后才执行函数, 确保获取到元素

第二步 在right-menu组件中

在created钩子中, 通过this.$nextTick(()=>{...}), 获取每一个li, 将他们的offsetTop减去上面的header和title的部分, 放到一个数组中, 并通过事件总线传到left-menu中

第三步 在left-menu组件中

1, 在computed中设置currentIndex属性, 通过比较pos.y和offsetTops数组确定左侧菜单应在哪一个li, 赋予currentIndex;

2, 在li中动态绑定:class=“{currentIndex == index}”, 此时计算属性currentIndex被使用, 并随着pos.y的改变不断刷新;

3, 在li上绑定点击事件, 通过foodScroll.scrollTo改变right-menu的滚动位置, 此时pos.y随之改变, 反向改变currentIndex, 被点击的li上绑定的class生效

**至此 大功告成**

3月27日第六天

数组务必用for of循环

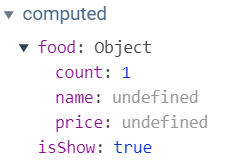
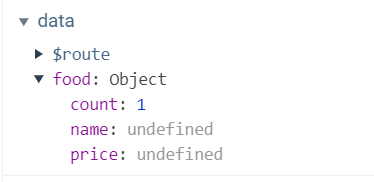
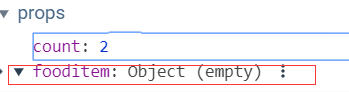
父组件向子组件传递对象数据时:

如果该数据在props中没有设置默认值{}, 在子组件的data中被使用时, 运行时会报undefined错误(不影响显示结果), 这是因为开始的时候数据还没有传递过来, 而在这之前开始了计算并赋值但是写在computed中就没有问题了(控制台不报错)

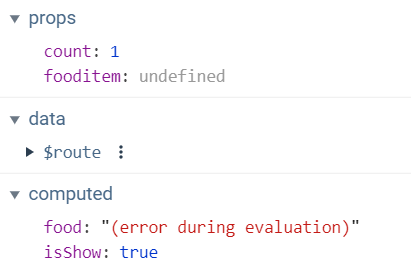
\*如果没有设置默认值, 而数据没有传递过来, 接收数据的变量为undefined, 也就是声明未赋值

\*\*data中的数据也是实时更新的, 随着依赖的数据变更而变更

设置默认值时的data和computed里在devtools的表现, 显示是一样的, 而且这个显示不会变化, 也就是数据的实时更新不会显示在devtools里, 只是显示了初始状态



未设置默认值时写在computed里, 也出了问题, 只是没有体现在console吧



**总结**: 这种情形的方案一是设置与数据类型匹配的默认值, 方案二是使用v-if=”数据”, 拿到数据才渲染

computed属性真的很好用 一是因为他有缓存 在数据不更改的情况下不必每次运行 二是它随着数据更新实时更新, 不断调用执行 实时响应变更 万分适用数据不断变化 并造成影响的情形

一个公有组件用在两个地方, 是创建了两个备份, 这两个备份之间毫无关联, 数据不共通不共享

要做到他们之间数据的同步, 除非他们使用的是一个对象类型数据, 他们的操作只改变其中的值

**具体问题具体分析**

解决这个问题的时候, 哪怕是类似的问题, 也要专注在这个问题上, 不要去想之前怎么做的, 专注本身的逻辑. 除非是完全相同的场景, 否则一点点差别也可能导致解决方案天差地别

要善于吸收每个人每个方案的思路和优点, 不要搞个人崇拜, 受某个人影响过重

再次强调 自己的思考 自己的思路

**购物车**

购物车这一块对数据的操作能力是有要求的, 思路不清楚, 容易在里面兜兜转转

之前做购物车时的两个不当之处:

抽出了商品部分要展示在购物车的属性, 这样有两个弊端, 一个是与原本数据的割裂, 二是可能当前没有展示的部分, 在以后在其他部分需要展示, 再去添加就被动了

在点击添加按钮时, 遍历收集商品的数组进行遍历, 通过id能判断是否已经添加过, 强烈感觉还是要**专注自身,** 其实可以给自己添加count属性, 通过这个属性的状态来判断, 总之, 操作一个数据时尽量减少对其他数据的依赖和影响

**以及再再次的强调 如果一个东西越想越乱了 甚至脑子也不转了 不是自己困了呆了 多半根本逻辑错了 不要拼老命去把毛线团解开 试试别的法子**

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

**对象中的属性更改后dom并不立刻更新, 通过保存部分刷新网页会更新**

**之前遇到的都是如何在dom更新后再操作的问题, 这初始受vue生命周期影响(要在合适的钩子写code);之后受dom异步更新机制影响(数据改变,对视图的更新已经蓄势待发,只等nextTick;**

**这次遇到了的是有些数据更改无法触发试图更新,除非强制刷新. 这是因为这些数据是非响应式的**

在vue中，改变某个对象的值时（如数组中的某一项或者对象的某个属性），vue并不会触发dom更新，这时就需要我们自己手动来操作。

原因：vue监听对象的变化，但是无法监听对象自身属性的改变

方法一：强制刷新 this.$forceUpdate();

此行为会重新渲染整个dom，除数据层次太多外，不建议使用

方法二：对于已经创建的实例，Vue 不允许动态添加根级别的响应式属性。但是，可以使用 Vue.set(object, propertyName, value) 方法向嵌套对象添加响应式属性。

但是, 如果我在当前组件的data中新建一个对象{count: 2}, 通过点击事件改变count, 不管我是通过计算属性转一下, 还是直接用data中的, dom都会更新, 这是为什么呢? 难道是因为不更新的数据是从父组件传过来的?不至于吧

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

洋洋洒洒一大篇, 查了好几篇文档, 但是发现与自己的试验相违背, 后来终于发现文档里说的都是错的

**以下为正确解答**

Vue.set()(vuex用, 因为它不是vue的实例, 所以不能用后者)/this.$set()

Vue.set( target, key, value )

向响应式对象中添加一个属性，并确保这个新属性同样是响应式的，且触发视图更新。它必须用于向响应式对象上添加新属性，因为 Vue 无法探测普通的新增属性 (比如 this.myObject.newProperty = 'hi')。注意对象不能是 Vue 实例，或者 Vue 实例的根数据对象。

\*\*通过直接赋值新增改变属性, 在devtools会有体现

原来这些都写在文档里了, 还是要过一遍文档 , 至少遇到坑的时候知道可能是哪里的问题, 做实验也最好有方向, 尽量避免瞎做(一个个试)

从官方资料学特性 从老师那里学应用

有些时候搞来搞去折腾来折腾去, 在网上查, 看到别人搞来搞去搞出来的文档, 觉得这个知识好神秘好奇诡好坑 最后发现其实都写在官方的文档里 正经的书里, 只不过是常规操作 万变不离其宗 所以如果坑了 要么是知识理解错了,要么是有知识盲区

3月28日 第七天

组件里只写组件自身 在父组件-wrapper写和其他组件的交互 如定位神马的 要让组件看起来像一个独立的可复用的 尽管并不会复用它 专注自身 干净利落 wrapper交互分工明确 逻辑清晰 切忌粘连不清 如果和兄弟组件交互 在父组件里处理更方便

昨天做这个还觉得不清不楚不知道如何做 今天赶紧更熟悉 更清晰 更顺手了

要努力跨过这段半生不熟的时期

功能一个个做, 一点点进阶, 步步检验 方便debug 渐进增强

子组件向父组件发送事件 接收要写在父组件里的子组件引用上, 想想父传子也是啊, 写在子组件引用上传过去的, 所以父子只是一个位置而已, 是子组件自身和子组件在父组件的引用之间的互传

**购物车详情页展示**

第一步 点击展示

当购物车logo中有货品时, 点击切换展示和隐藏, 通过条件判断和改变v-show的值实现

第二步 从下方往上走

通过绑定class改变bottom的值实现, 点击事件中改变class的状态

\*相对于父元素定位, 并作为第一个子元素, 这样它的出生地在最前面

第三步 transition

问题: 因为详情页的高度是可变的, 我知道bottom显示时的位置, 但是不知道初始的位置, 所以写了一个较大的值, 这样的话, 连续点击时会有问题, 表现为延迟

(看了一下现在的饿了么, 购物车是有一个最大高度的, 超过了就需要使用滚动来查看所有商品)

\*虽然要transition的元素是包裹在transition组件内, 但是transition类名和元素类名要写在同级, 不要写成父子

\*\*写完transition类名设置就可以产生动画了, 但是要在动画完成之后还保持状态, 要把这个状态写在元素类名下, 动态绑定类名到元素

\*\*\*enter是状态的改变, leave是状态的还原, 不要搞反了

问题:

点击隐藏时bar会闪动

解决方案一: 把v-show换成v-if

解决方案二: 把详情页z-index设为-1

(大概知道为什么, 大概知道该这么解决, 具体为啥不知道)

哦~大概是因为这个详情页绝对定位之后是在bar上方的, 所以隐藏时, 是先覆盖上方, 然后隐藏掉了, 这个过程很快, 体现为闪动

第四步 在发现bug后产生的需求

购物车清空后, 详情页不会消失, 因为它的消失与出现是由点击事件控制的

解决方案: 通过computed做一层中转判断实时改变v-show状态, 如果不等于0就返回点击事件改变的值, 如果不是就返回false, 这样将两种情况结合在一起: 点击购物车是不为0的情况下, computed监测是纳入购物车从有货变成0的情况. 不过呢, 因为动画还是会有延迟, 或许不需要这个退出的动画

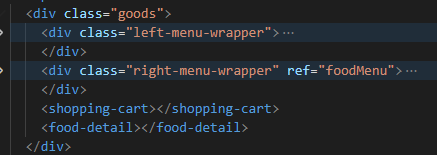
(现在是摸索实现后总结思路 后面应该要先写思路后实现)

以前没来的及写, 现在要把流程写下来, 写下来才能反思和提升, 要有意识去学习, 随缘可要不得

商品详情页

这个详情页每个商品都要有, 但是一次只会展示一个, 因此只需要使用一次这个组件即可, 食品信息在点击事件发生时传出, 这样每次渲染都是不一样的

预备



第一步

在right-menu组件中设置商品点击事件, 并携带商品信息发送给父组件, 然后通过data 传到food-detail组件

!复制粘贴时一定要特别小心

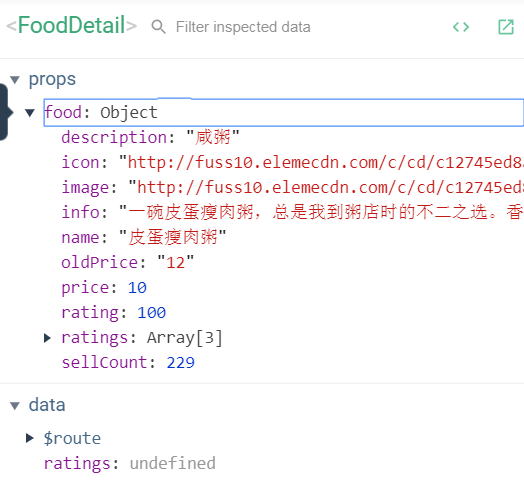
强烈的学技术的冲动, 激动的对未来的憧憬, 转行附加的过度期望和过度担忧在我心里斩断, 只有纯粹的开始新篇章做一份让自己愿意不断学习进步的工作的期盼和喜悦

评论部分

场景: 将传到子组件的food中的ratings赋值给data中的ratings作为初始状态, 设置点击事件改变ratings来切换展示评论类型

问题: 点进详情页的时候为空, 需要点击选项卡来展示;

(devtools中看到, 此时props中拿到了数据, 但是没有赋值, ratings为undefined)



同时, 当退出一个商品的详情页进入另一个商品的详情页的时候上一个商品的评论展示会保留, 点击之后才会刷新

原因: 共用一个组件, 由于没有赋值, 依然是之前的数据

**问题总结: 如果将子组件上的数据赋给data 然后渲染，数据不能动态更新。**

**解决方案: (通过合理的关键字搜索到, 所以说不要自己苦想)**

使用watch来监听props中的变化, 一旦发生变化, 立刻赋值到data中

vue中watch的几种用法:

<https://blog.csdn.net/wangbinXMU/article/details/97619725>

3月30日 第八天

详情页点击加入商品到购物车

点击切换评论类型时的class变化

评论类型与手的图标关联

首页商品页面内容被购物车栏目遮挡

回顾一直遇到的问题 很多都是数据触发视图更新的失败

第一 只有响应式数据能够触发视图更新 某些数组和对象方法改变的数据是非响应式的, vue提供了很多变异方法来制造响应式

第二 在响应式数据触发视图更新的过程中有几个问题:

在使用时请求或者传递的数据还没有拿到, 为undefined;

dom更新是异步的, 并不是立即更新, 因此紧接着的dom操作会出错;

静态存储器赋值不是实时的, 将props中的值赋给data不会立刻生效的, 在methods和watch等动态中可以

\*data和computed中的值都是实时更新的, 区别在于data要不断赋值, 它不能自动,适用于事件; 而computed随着依赖值的更新自动就更新了, 适用于监测, 平时那种是省略写法,用的是getter, 没有setter方法, 是不能赋值的

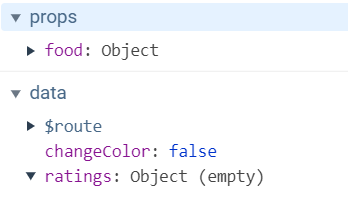
评论的切换和筛选出现问题, 一步步分析, 分离问题然后逐一解决, 盲目去看视频求助真的要不得的

问题一: 退出后重新进入同一个商品的详情页, 评论数据还是退出时候的数据

原因: 详情页设置是v-show, 在传入数据未发生改变的情况下, 视图不变, 维持在之前已经变更的状态

解决方案: 改为v-if

但是呢, 又产生了另外的问题, 没有数据展示, 从devtools可以看到经接收到父组件的数据, 但是通过watch动态赋值给data中ratings还没成功, v-show的时候是成功的, 按理来说, 都是通过watch监测然后赋值, 为什么会有区别的, 不知道, 还是先用v-show吧



问题二: 当选择的是”全部”时, 选中了”只看内容”, 但是切换到其他选项时, 就和没选中一样了, 逻辑有问题

两个事件的状态同时作用于一个数据的时候的逻辑问题

解决方案: 用两个变量保存在data里, 然后进行if判断来选择使用哪一个数据

但是感觉不怎么优雅啊emmm

三个选项 全部 满意 不满意

叠加状态 只看有内容的

两个数据 一个是全部评价 一个是经过筛选的有内容的评价

问题 选择只看内容后退选 状态回不去

难道要三个数据???

以满意为例分析所有的状态

点击了满意后 再点击只看内容 没问题

再次点击只看内容 有问题 状态已经改变 回不去了

点击只看内容后 在点击满意 没问题

再次点击只看内容 有问题 状态已经改变 回不去了

结论: 似乎还需要一个来保存点击只看内容之前的状态, “全部”选项是有的, 但是”满意”和”不满意”没有, 需要再建一个变量来保存

更不优雅了orz

增加了变量 出现问题

最开始没有点击三个选项的时候这个数据是空的...就没有内容展示了

原因: 初始值设置为空对象

解决: 在watch里把全部评价赋值给它

最后的问题, 由于数据是在选择某个选项后保存的, 如果选择了一个选项后, 选择另一个选项, 在选择原来的选项, 那么点击 只看内容 数据就有问题了 似乎, 好像, 需要单独建数据来保留他们的状态...看看视频有没有好办法, 这个时候时机才对(带着问题去找答案)

v-show还能写函数, 当然能啊, 只要最终的值是布尔值就可 还是要多看涨姿势啊 按理来说不至于搞那么多数据的

通过一个函数多个参数来一次性判断, 而不是像我之前的办法那样, 通过多次判断选出需要展示的数据, 而且这样做ratings一直没有改变过, 如果不是永久改变一个值, 那么最好不要用事件来回改变它

Vue是允许js表达式的, 在dom的绑定数据中, 在script标签里, vue是三剑客的陈列柜

V-show可以与v-for一起使用, 选择性地展示部分数据

按条件过滤数据然后渲染---->渲染的过程中按条件过滤

\*\*\*全部 不需要筛选 直接true 虽然是三个按钮, 但是只有两个types--满意和不满意

问题: 给选项卡添加选中时的状态后, 点进不同的商品详情页它的状态会保留

原因: ~~同一个组件, vue的复用渲染~~ 同一个组件, 只是数据不同, class的状态还在上面呢

解决: 进入不同详情页时初始化, 写在watch里, 随 food数据更新而改变

**按老师的思路修改完毕**

**至此 评论选择与筛选功能完成 欧耶**

做自己觉得对的事, 做自己愿意做的事, 不要用高道德要求自己, 不要屈从于任何人的压力

无论结果如何 不指责自己 只总结经验 汲取经验教训 继续前进

如此 方可免于后悔和怨恨

终于开启了早晨学习 明天要起的早一些

如何破除层层迷障 获得真正的勇气

**要注意细节鸭**

**中间插入**

问题: 发现首页菜单点击加入购物车的加号按钮, 会进入详情页, 这显然不是想要的效果

原因: 很简单, 事件冒泡嘛

解决: stopPropagation

\*stopPropagation与stopImmeidatePropagation的区别

前者阻止当前事件结束后父元素上该类事件的触发; 后者阻止当前事件结束后本元素上其他同类事件和父元素上该类事件的触发(话说都是点击事件, 怎么知道谁先谁后)

在当前情景中, 加号按钮自身也有点击事件, 那么在这个事件处理函数中使用stopPropagation即可

之前也看了这两个的区别, 好像懂了, 又好像没懂, 因为自己没有”正向”用过, 也就是不知道它的使用场景, 不使用, 不知道使用场景, 不叫会一个东西, 总觉得朦朦胧胧

抽离rating-select

\*\*\*要早早观察好要复用的组件, 以免写好了再抽离

\*如果自定义事件只传递了一个参数, 在dom中不要写上, 否则会报错

\*\*我是这样做的: 父组件中定义数据, 子组件发出事件, 父组件接收并改变数据, 然后传递给子组件(即时地), 这样做的时候要注意在watch中对数据进行初始化 以免进入另一个商品详情页的时候还保留着之前的状态

感觉抽的还不够 应该把下面的评论展示也带上 把具体展示的内容留插槽

\*在子组件中留了插槽, 在父组件中用内容填充插槽, 插槽内容是无法访问子组件的数据的, 简单粗暴一点说, 就是写在哪里就访问哪里的数据, 否则就能靠传

**父级模板里的所有内容都是在父级作用域中编译的；子模板里的所有内容都是在子作用域中编译的。**

\*\*数据的话, 写在哪里就用哪里的, 但是css样式, 写在哪儿都可以, 写在子组件里也行, 写在父组件里也行(注意: 子组件插槽部分的class样式可以写在子组件里, 也可以写在父组件里; 子组件的class样式只能写在子组件里, 但是class名还是可以在父组件里用, 作为插槽class的父级)

插槽可按阶段分为

匿名插槽: 一个子组件中只能有一个, 在父组件中填入的内容会填充(不需要根标签)

具名插槽: 可以有多个, 可与匿名插槽共存, 在向具名插槽提供内容的时候，可以在一个 <template> 元素上使用 v-slot 指令，并以 v-slot 的参数的形式提供其名称, 将内容放入其中即可

作用域插槽

OK 抽离成功了 明天写起评价页就很方便了

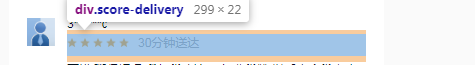
三月三十一日 第九天

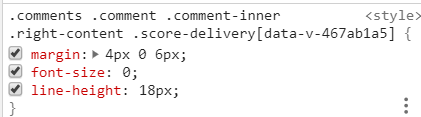
inline-block为什么没有对齐而是换行



inline-block默认的宽度是auto，意思就是宽度随内容增加而增宽，随浏览器宽度而换行。不设置宽度，会根据内容的情况撑开。内容不够，还能一行，内容足够的情况下，会占据100%的宽度，那时就换行了

不要把行高写在父元素中, 不知道这是为什么





行高只能让文本垂直居中对齐的, 并不能让非文本元素垂直居中

文本超出部分显示省略号

            overflow hidden

            text-overflow ellipsis

            white-space nowrap

四月二日 第十天

重构完善”饿了么”外卖

**预备工作**

**设计文件结构**

删除自带的各种东东;

引入css重置文件, 引入自定义字体文件;

在views下建立四个一级路由文件夹;

建立vue.config.js, 配置别名

\*\*\*配置完成后需要重启项目, 否则不生效报错



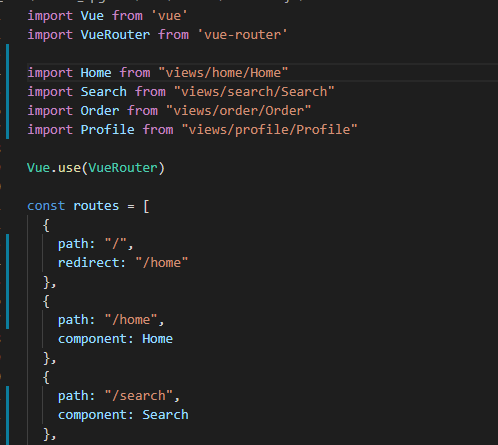
安装各种依赖 stylus stylus-loader axios

npm install stylus stylus-loader --save-dev

<style lang="stylus" rel="stylesheet/stylus" scoped></style>

npm install axios

**部署路由**



**部署数据接口**

**第一种方法**: 直接写在原型链上(简易用法, 有弊端)

//引入axios

import axios from "axios"

axios.defaults.baseURL = "http://106.12.11.162:8888/api/private/v1/"

Vue.prototype.$http = axios

向服务器发送表单

this.$http.post("login", this.loginForm)

//”login”为子接口名称; this.loginForm为element-ui绑定的表单对象

login(){

      this.$refs.loginFormRef.validate(valid => {

        if(!valid) return;

        const result = this.$http.post("login", this.loginForm).then(res => console.log(res))

        console.log(result);

      })



(请求返回一个Promise {pending}, 很快赋值并打印; 然后拿到数据才执行打印数据的回调函数)

--使用async await进行优化, 结果是一样的

login(){

      this.$refs.loginFormRef.validate(async valid => {

        if(!valid) return;

        const result = await this.$http.post("login", this.loginForm)

        console.log(result);

      })



Promise深入理解: <https://blog.csdn.net/qq_27626333/article/details/76223842>

**第二种方法**:

(是否可以复制一份作为模板以后用呢...开始先自己建吧)

Vue的根实例在main.js中

