4/17

先学习html,css

总结：

这一天下来大致把html介绍看完了，css看了一部分，看了js的简介。

大致学习的内容有

* html元素的内容，区块级和行内元素的差别，html文件的元素组成。
* Head标签的功能（存放不直接展示给网页浏览者的信息）
* Html的列表、行级强调、下划线
* 超链接(href,title和target的使用)
* 图像的打开
* 学习了css的语法结构以及selector的概念(.对应着class，#对应着id)

其中，Advanced text formatting这一章节看得不是很懂，有需要再回来复习

4/18

重点学习js，而html和css只需要大概了解

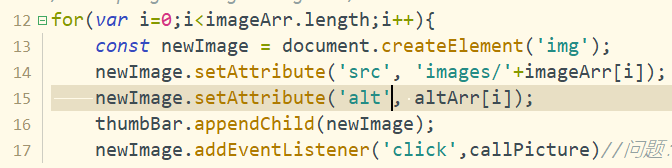
总结：

js看到了对象的部分。明天的任务是写掘金的静态页面。

大致学习的内容有：

* 变数的概念(let局部,var全局声明)一般推荐用let，但是网页练习里用var居多
* Js很不一样的一点是定义可以在声明之前，但是一般不推荐
* Js只有number一个数字类型
* 字符串（模板字符串``的使用（${表达式}））
* Number以及toString函数（数字字符串转换）
* Replace,slice,indexOf方法（字符串方法）
* Array操作（split,join,push,pop,shift,unshift）
* 条件，循环，函数，回传值基本和其他语言差不多

遇到的bug：一开始写的是callPicture(event)，直接把参数也写上了导致传得是结果，而不是参数。



4/19

今天把css内容大致学习一下，下午开始写掘金的静态页面。

总结：

大概的过了一下css的知识点

学会了css选择器的概念和分类，以及层叠和继承

* 层叠：

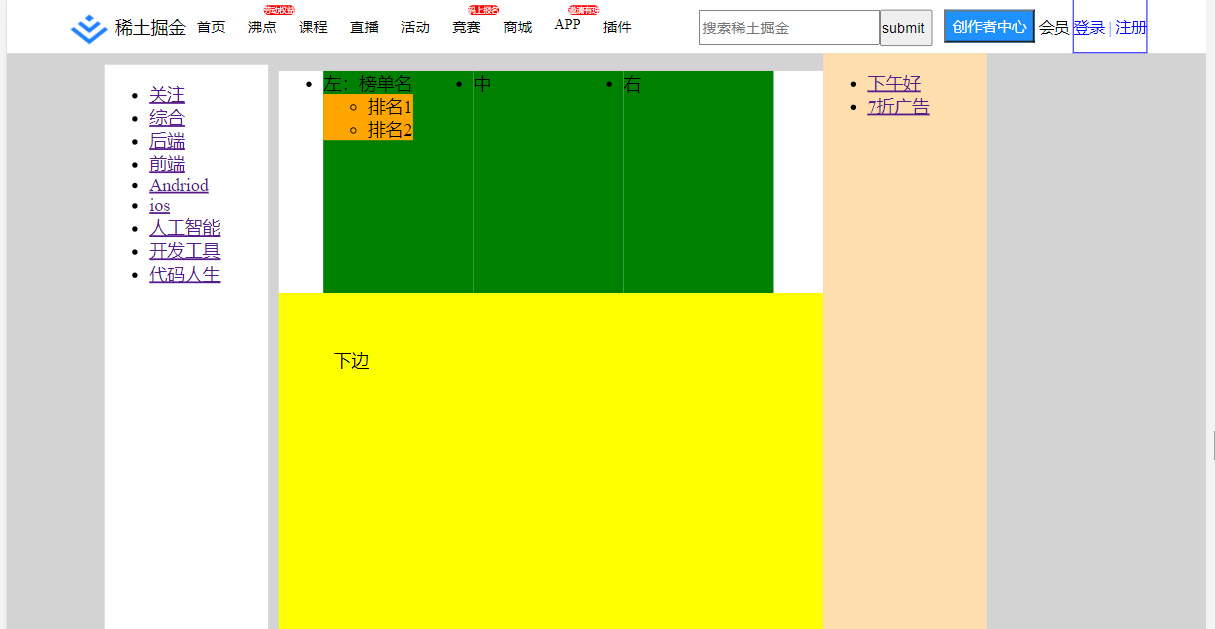
资源顺序：后者覆盖前者

优先级：内联（不推荐，在style中）>ID>类（类选择器，属性选择器，伪类）>元素（元素、伪元素选择器）

* 覆盖：用户代理样式表常规说明（浏览器默认样式）<用户样式表常规说明（用户自定义）《作者样式表（web开发人员）<

作者样式表中的!important<用户<用户代理

弹性盒子display:flex用来排列元素。



4/20

继续编写静态页面

总结：

大概把网页的排版搞定了。遇到的问题，外边距覆盖的概念给忘了，找了很久才找到冲突点。Display:flex的使用大概掌握，通过flex-direction能调整主轴方向，justify和align一个控制主轴一个控制交叉轴，item和content这两个暂时分不清，值有space-around，space-between和center，start，end



4/21

添加js，学习js

总结

分析待添加功能：

点击搜索栏，搜索栏扩大覆盖掉创作者中心，并更新placeholder，点击其他部分回复(未添加)

网页下移时，搜索栏固定（添加，css更改position:fixed)

底下的推荐和更新栏目的切换（已添加，两种实现，第一种直接用js中存放推荐和更新栏目的内容，并用innerHtml来覆盖ul里的数据，第二种直接将数据保存在html中，使用js来更改cs中他们的display属性，需要注意的是赋值要加上’’，比如展示更新栏目时，需要将推荐栏目的display设为’none’，并将更新栏目的display设为’block’）

鼠标移动到登录/注册下跳出栏目（未实现）

Js学习内容：对象基础，OOP，继承都和c/c++差不多，只是用的语法有些许不一样

新的知识点：对象原型，JSON数据的使用，然后掉回头把html的图片，视频和音频内容看了一点但是没看完。原型的概念还不是很深，

Object.getPrototypeOf()函数查看原型

设置原型的两种方法：

Object.create()：创建原型来创建一个以 personPrototype 为原型的新对象

使用构造函数：Object.assign(Person.prototype, personPrototype);将 personPrototype 中定义的方法绑定到 Person 函数的 prototype 属性上。

4/23

先暂定继续学习js吧

总结：

重新过了一遍，原型，感觉他和java的接口差不多，不过不一样的是接口需要自己实现，而原型是直接复用该函数。

异步就是开线程，让函数开始操作并立即返回不等待该运行结果，而是直接返回先干别的事。

回调函数则是一个被传递到另一个函数中的会在适当的时候被调用的函数。

Emm突然发现了用reference进的话js的内容有点差别，现在看回去补上知识点

函数提升，未声明就先用是可以的，不过定义的数据不会前移，以undefined代替。函数提升也一样。

ReferenceError表示不存在声明，undefined表示未定义（可能是提升）

Const不能通过重新赋值改变（对象属性和数组内容不被保护）

字符串转数字出了Number（）还有parseInt和parseFloat

数组面值种的多余,,会被认为是填充undefined

Undefined->false(boolean)

不合法的对象属性名，它必须用""包裹，访问其属性值，而是通过类数组标记 ("[]") 访问和赋值，而不能用.来访问

闭包是一个可以自己拥有独立的环境与变量的表达式（通常是函数）

内部函数只可以在外部函数中访问。

内部函数形成了一个闭包：它可以访问外部函数的参数和变量，但是外部函数却不能使用它的参数和变量。

函数的实际参数会被保存在一个类似数组的 arguments 对象中。使用arguments[i]获取更多参数

函数参数->剩余参数：允许将不确定数量的参数表示为数组function multiply(multiplier, ...theArgs)

每一个函数里都定义了自己的this方法，因此嵌套函数中，this.age会被内层的this.age给覆盖而导致代码逻辑出问题。

使用箭头函数 函数名(()=>{this.age})这样的话this.age就可以是外部函数的了。

对于更复杂的赋值，解构赋值语法是一个能从数组或对象对应的数组结构或对象字面量里提取数据的 Javascript 表达式。var foo = ["one", "two", "three"];

var [one, two, three] = foo;

4/24

学习js异步

总结：

今天复习了js的继承和原型链，加深了理解，还学习了异步的使用，并且编写了第二个页面以及第三个页面的部分。

4/25

继续学习js，并且完成了第三页面的编写工作，添加了鼠标悬浮切换图片的功能。

4/26今天卡了很久原因是css架构出错了

要使用left属性需要将其父容器设为relative或者absolute

外层边框固定大小，并且设置oversive:hidden就可以把超出的部分显示隐藏，内层图片列表宽度要加长来存放图片，最后是图片实例紧密连接排放

4/27

symbol的理解：

每一个symbol都是独一无二的

Symbol("foo") === Symbol("foo"); // false

因此用作对象属性名（防止属性名称冲突）

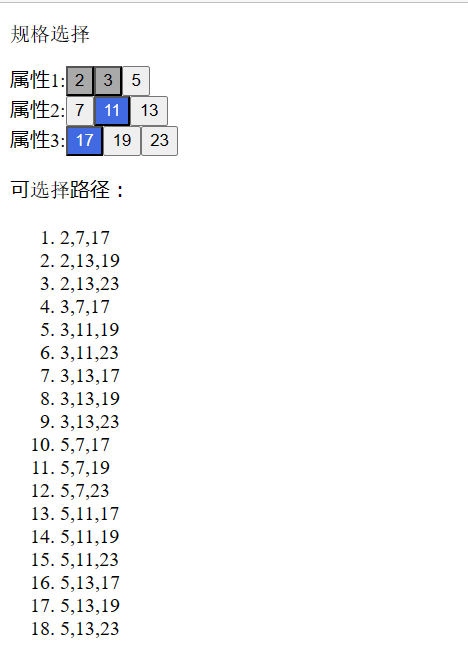
js对象没有私有属性，但为什么要有getter和setter呢？

答：不用的话是直接调用函数，用了的话调用属性直接就可以获取到。

4/28

编写sku算法

结果展示



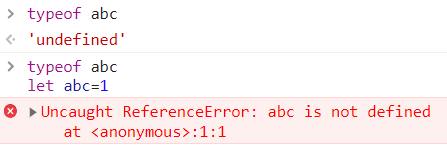


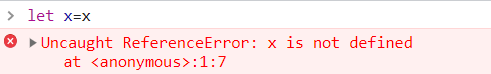
4/29~5月3放假

5月4-5学习ecma6

Ecma6可以通过babel插件切换成5

let命令：不提升，所以存在暂时性死区(死区内-referenceError，未声明或声明未初始化-undefined，死区导致typeof不再安全)，不允许重复声明





块级作用域：允许在块级作用域内声明函数。函数声明类似于var，即会提升到全局作用域或函数作用域的头部。同时，函数声明还会提升到所在的块级作用域的头部。（建议块内使用函数表达式而不是声明）

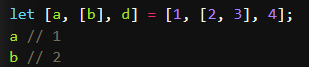
const 命令（不提升，死区存在，不可重复声明，指向的内存地址苏保存的数据不变（符合类型如对象数组数据可变））

顶层对象的属性（浏览器环境-window对象，Node -global对象。var命令和function命令声明的全局变量，依旧是顶层对象的属性（可用window.变量来调用）；另一方面规定，let命令、const命令、class命令声明的全局变量，不属于顶层对象的属性。）

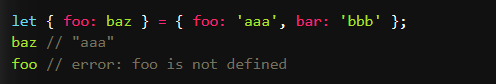
globalThis 对象（用来获取顶层对象，指向全局环境下的this）

变量的解构赋值（模式匹配，格式相同才能获取）

数组的解构赋值（次序匹配，解构失败=undefined，不完全解析只匹配一部分，只要有iterator接口就可用）



对象的解构赋值（变量属性同名匹配，不同名的话可用下图的形式，foo是模式，baz才是变量，解构赋值可以取到继承的属性）



字符串的解构赋值（被转换为类似数组的对象，有length属性）

数值和布尔值的解构赋值（右边是数值和布尔值，则会先转为对象，undefined和null无法转为对象会报错）

函数参数的解构赋值（undefined触发默认值）

圆括号问题（可用：赋值语句的非模式部分；不可用：变量声明语句、函数参数、赋值语句的模式）

用途（交换变量的值、从函数返回多个值、函数参数与变量名对应起来、提取 JSON 数据、遍历iterator接口对象、函数参数默认值）

变量解析

字符串

Unicode表示只适用于\u000-\uFFFF之间，大于\uFFFF的码点需放入大括号

大于FFFF的只能用for...of取

模板字符串（用反引号表示、${}内调用表达式、函数和变量，模板字符串可嵌套）标签模板

String.fromCodePoint()识别大于0xFFFF的字符

String.raw()返回一个斜杠都被转义（即斜杠前面再加一个斜杠）的字符串

includes()：返回布尔值，表示是否找到了参数字符串。

startsWith()：返回布尔值，表示参数字符串是否在原字符串的头部。

endsWith()：返回布尔值，表示参数字符串是否在原字符串的尾部。实例方法：repeat()

padStart()，padEnd()补全字符

trimStart()，trimEnd()删除空格

matchAll()匹配

replaceAll()替换

at()查询字符

RegExp

原本：

参数：字符串（第二参数为修饰符）

参数：正则表达式（不允许第二参数，返回原有表达式的拷贝）

Es6:

正则表达式允许第二参数，忽略原有的修饰符。

string方法在语言内部调用regexp的实例方法。（match,replace.search,split）

正则扩展的部分太杂了，剩下的部分就没仔细看了

数值

Math.trunc去除小数点

Math.sign返回+1,-1,0表示正负数

Math.cbrt计算立方根

函数length属性只返回到无默认值的个数

函数参数x=x会形成单独作用域，执行的是letx=x所以死区

Rest参数(...变量名)不需要arguments对象，相对也更加简洁（arguments对象是类似数组的对象，而rest参数得到的是数组），rest参数只能是最后一个参数，并且length属性不包括rest参数

函数参数使用了默认值、解构赋值、或者扩展运算就不能显式设定为严格模式。规避方法，全局严格以及函数包裹在无参数的立即执行函数里。

函数name属性返回函数名，Function构造的实例返回”anonymous”，bind返回的函数name前加上bound

箭头函数，圆括号内参数形式如下：

(参数)=>返回值

(参数)=>{函数体 内部要有return 返回};

要想返回对象需要在对象外加括号，否则会被解释为代码块。(参数)=>({对象内容})

作用：简化回调函数

（1）箭头函数没有自己的this对象。（this指向上层作用域的this）

（2）不可以当作构造函数，也就是说，不可以对箭头函数使用new命令，否则会抛出一个错误。

（3）不可以使用arguments对象，该对象在函数体内不存在。如果要用，可以用 rest 参数代替。

（4）不可以使用yield命令，因此箭头函数不能用作 Generator 函数。

尾调用 =函数最后一步直接返回另一个函数的调用，优化后只保留内层函数的调用帧，从而节省内存。（只有不再用到外层函数的内部变量，内层函数的调用帧才会取代外层函数的调用帧）

尾递归同理，优点不会出现栈溢出，因为只有一个调用帧。调用复杂度(O(1))

2017后允许函数参数最后带逗号。

Catch函数允许不带参数。

数组

扩展运算符(...)，rest函数的逆运算，将数组转为用逗号分割的参数序列，用于函数调用（rest函数用于函数参数声明，将参数转换为数组）（注：只有在函数调用时，扩展运算才能放在圆括号中）

不再需要用apply函数来将数组转为函数参数。

Array.from()将数组对象和可遍历对象转为数组

Array.of将一组值转为数组，用于代替Array，因为Array(3)会被解释数组长度为3的数组。

实例方法

copyWithin(target, start = 0, end = this.length)

从start到end的数据替换掉target开始的数据。

find()方法的参数是回调函数，找到第一个符合要求的数据并返回值。同理，还有findIndex()用于返回索引，findLast以及findLastIndex方法则是逆序查找

Fill使用给定值填充数组，2，3参数指定起始和结束位置。

Entries,key,values=>键值对，键名，键值

include方法返回bool,判断数组是否包含给定值。

Flat,flatMap方法将数组拉平成一层，后者接受一个参数做map操作。不过后者只能展开一层。

group()的参数是一个分组函数，分组函数可以接受三个参数，依次是数组的当前成员、该成员的位置序号、原数组

groupToMap使用方式一样不够返回值是map结构。

对象

简写的对象方法不能用作构造函数

Es5:

for...in循环：只遍历对象自身的和继承的可枚举的属性。

Object.keys()：返回对象自身的所有可枚举的属性的键名。

JSON.stringify()：只串行化对象自身的可枚举的属性。

Object.assign()： 忽略enumerable为false的属性，只拷贝对象自身的可枚举的属性。

Es6:

（1）for...in（同es5，不含 Symbol 属性）

（2）Object.keys(obj)返回一个数组，包括对象自身的（不含继承的）所有可枚举属性（不含 Symbol 属性）的键名。

（3）Object.getOwnPropertyNames(obj)

Object.getOwnPropertyNames返回一个数组，包含对象自身的所有属性（不含 Symbol 属性，但是包括不可枚举属性）的键名。

（4）Object.getOwnPropertySymbols(obj)

Object.getOwnPropertySymbols返回一个数组，包含对象自身的所有 Symbol 属性的键名。

（5）Reflect.ownKeys(obj)

Reflect.ownKeys返回一个数组，包含对象自身的（不含继承的）所有键名，不管键名是 Symbol 或字符串，也不管是否可枚举。

对象解构：{x,y,...z}={x:1,y:2,a:3,b:4}

对象扩展:z={a:3,b:4},n={...z}//...用于取出参数对象的所有可遍历属性，拷贝到当前对象之中。

数组是特殊的对象，所以对象的扩展运算符也可以用于数组。！！！

Object.is()比较两个值是否相等

Object.assign()用于对象的合并（浅拷贝，同名属性替换不叠加，数组视作对象相同索引替换，若是取值函数取值后复制而不是函数）

Object.getOwnPropertyDescriptors()返回某个对象属性的描述对象（descriptor）

\_\_proto\_\_属性，Object.setPrototypeOf()，Object.getPrototypeOf()

Object.keys()返回一个数组，参数对象自身的（不含继承的）所有可遍历属性的键名，Object.values()，Object.entries()同理

Object.fromEntries()是entries的逆操作，将键值对数组转为对象

Object.hasOwn()判断是否为自身的属性

set-add,delete,has,clear,keys,values,entries,forEach，forOf

weakset成员只能是对象，其中的对象都是弱引用（即其他对象不引用就会被回收，不管weakset）。weakset没有size属性无法遍历。

Map也很熟悉了。

Map->数组[...myMap]

数组->Map new Map(数组)

Map->对象，for..of取出即可封装成对象。

对象->Map new Map(Object.entries(obj))

Map->JSON JSON.stringtify(对象)/JSON.stringtify([...map])，前者对于字符串，后者为非字符串，改用数组

JSON->Map JSON.parse()得到后再转换为Map

WeakMap只接受对象作为键名，并且都是弱引用

WeakRef是用于直接创建弱引用，感觉暂时用不到就不看了。

Proxy

在目标对象之前架设一层“拦截”，外界对该对象的访问，都必须先通过这层拦截，因此提供了一种机制，可以对外界的访问进行过滤和改写。

方法

将类的派生视作一棵树，以object为根，向下蔓延开来

协变-允许使用更具体的派生类类型<?extends fruit>

逆变-允许使用更抽象的祖先类类型<? super orange>

协变限制了祖先类（上限），所以可以安全读出(get)，用作producer

逆变限制了子类（下限），所以可以安全香气中写入(set)，但是取出时类型无法确定所以get不安全

生产与消费可以这样对应: Producer-Extends(协变get), Consumer-Super(逆变set)。

因此函数的输入参数大部分都是协变，返回结果都是逆变。

Module

Export 模块对外接口,import输入其他模块的功能

Export是动态绑定的，可以实时获取块内部实时的值。

import是静态执行，所以不能使用变量以及表达式

import会执行所加载的模块，多次import指挥执行一次。

与之相对的commonjs用的require命令

Export ...from是import和export的符合写法，表示转发，并没有导入模块中因此不能再当前模块中直接使用。

Module的加载

CommonJS 模块输出的是一个值的拷贝，ES6 模块输出的是值的引用。

CommonJS 模块是运行时加载，ES6 模块是编译时输出接口。

CommonJS 模块的require()是同步加载模块，ES6 模块的import命令是异步加载，有一个独立的模块依赖的解析阶段。

5/6

学习ts，ts是对js的补充需要使用tsc来进行编译编译后相对与js，加上了类型的检查。

5/8

继续学习ts

5/9-5/11

学习react

Jsx（可以让你在 JavaScript 文件中书写类似 HTML 的标签。）

只能返回一个根元素，多个的话需要使用一个父元素包裹起来

2. 标签必须闭合

3. 使用驼峰式命名法给 所有 大部分属性命名！

JSX 中通过大括号使用 JavaScript

对象的话需要使用双大括号

周2-4react

5/12

周5webpack+开会分享软件工程

Webpack是js的静态模块打包工具，处理应用时会在内部从一/多个入口点构建以来图，并将所需的模块组合成一/多个bundles（静态）。

入口-entry

输出-path,filename

Loader(webpack只懂js和json)，让webpack处理其他类型文件并转换为模块,test识别待转换文件,use定义转换使用的loader。

Plugin用于执行范围更广的任务，如打包优化，资源管理，注入环境变量，使用插件=require()+添加到plugins数组（new创建实例）

Npm install --save用于打包到生产环境，--save-dev打包到开发环境

5/15

Webpack+cli+button

先用webpack搭建，然后开始

ESlint：代码检查工具； .eslintrc.js

prettier：一个 Opinionated 的代码格式化工具； .prettierrc

安装eslint和prettier教程https://zhuanlan.zhihu.com/p/62401626

任务要求：编写组件库以及静态button组件

根据所给的参考网站开始学习怎么编写，看了一下架构是vue+varlet的，一开始毫无头绪要从哪里下手，就先开始快快地看了一遍源代码：vue3的代码被分为不同的库存放在根目录上的packages里，大致了解后只知道可以参考的代码在vue-cli下以及varlet-ui下，前者大概是说怎么配置开发环境和生产环境，后者存放的是具体的组件。其他看起来比较需要的还包括varlet-use但这是存放动态组件效果的库，暂时不用搭理。

[面向 Vue3 的 Material 风格移动端组件库 (gitee.io)](https://varlet.gitee.io/varlet-ui/" \l "/zh-CN/home)

[Vue组件库 | 如何从0到1开发一个开源组件库 - 掘金 (juejin.cn)](https://juejin.cn/post/7010553100222070792)

看了上诉网页，得知开源组件库需要提供的功能有哪些，首先是需要选定相应架构和设计系统。Vue3使用的是monorepo架构和material design。

我们采用了拆包的架构, 主要是通过yarn workspace和lerna实现，好处在于我们可以把通用的依赖都做成一个包进行单独发布，在构建组件库的过程中也可以同时产出一些实用的工具，也为后期项目的扩展打下了基础。同时lerna有着完善的发包机制，让我们不需要太关心包和包之间的依赖关系。组件库则设计成其中的一个子包，所以Varlet在未来可能不会仅仅是一个组件库，随着包的增多可能会变成一个解决方案，实际上我们也正在朝这个方向探索。

Material Design 引入z轴概念，也就是通过对组件的阴影加深来产生前后层次感。

yarn工具的安装与开启

16.10开始就内置在node.js里，管理员模式下使用命令corepack enable即可激活。

[(62条消息) yarn、npm、cnpm三者如何优雅的在一起使用 ？\_yarn npm 同时使用\_hyupeng1006的博客-CSDN博客](https://blog.csdn.net/hyupeng1006/article/details/93881364)

Yarn和npm的差别

Yarn并行安装，效率更高而且输出更加简洁。

Yarn global install add

Add等价于npm 的--save

package-lock.json 不应写进 .gitignore

[从 0 构建自己的脚手架/CLI知识体系（万字）](https://juejin.cn/post/6966119324478079007" \l "heading-10)

脚手架的雏形-创建命令-》询问问题-》生成项目文件

创建：npm init(生成package.json)->创建入口文件（修改package.json)->npm link

Npm link链接到全局后像删除可以找到node.js的安装地址进行文件删除。

询问问题：inquirer包可以设置问题(prompt)，ejs模板可以将用户输入数据渲染到模板文件上

这时候还不是很懂vue和varlet是做什么的，只是过去的项目中用过vue所以对他的印象只是在一个文件里写有html,js以及css。

[前端如何搭建一个简单的脚手架 - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/143520044)

还是不清楚要干什么，暂时只知道脚手架的目的就是简化搭建流程吧？也就是写一个可以让人直接git clone的组件库？

[如何搭建组件库-基础实战篇？ | 人人都是产品经理 (woshipm.com)](https://www.woshipm.com/pd/3290943.html)

还是不清楚要怎么搭建组件库，继续搜索

[练练手吧！ TS + React 小项目（01 创建项目） - 掘金 (juejin.cn)](https://juejin.cn/post/6956112714779328519)

网页使用了create react app

官方支持的创建 React 单页应用程序的方法，它提供了一个零配置的现代构建设置。

[arronKler/lime-ui: 《如何打造一套vue组件库》示例代码 (github.com)](https://github.com/arronKler/lime-ui)

[react-component/ at master · Aus0049/react-component · GitHub](https://github.com/Aus0049/react-component/tree/master)

[course-dasheng/umaso: 一个用来学习的React组件库 (github.com)](https://github.com/course-dasheng/umaso)

[React 组件库搭建指南（一）：项目初始化 · Issue #3 · worldzhao/blog (github.com)](https://github.com/worldzhao/blog/issues/3)

以上是那时候找的一些组件库参考

[【架构师（第八篇）】脚手架之 commander 框架使用方法 - 掘金 (juejin.cn)](https://juejin.cn/post/7081800378605371406)

[2022代码规范最佳实践（附web和小程序最优配置示例） - 掘金 (juejin.cn)](https://juejin.cn/post/7056948788329447454)

[使用ESLint+Prettier规范React+Typescript项目 - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/62401626)

看vue源码时的参考

[(64条消息) 三.派发与广播-自行实现 dispatch 和 broadcast 方法\_Lucky.cs的博客-CSDN博客](https://blog.csdn.net/weixin_48639457/article/details/129666813" \l ":~:text=%E5%9C%A8 Vue.js 1.x %E4%B8%AD%EF%BC%8C%E6%8F%90%E4%BE%9B%E4%BA%86%E4%B8%A4%E4%B8%AA%E6%96%B9%E6%B3%95%EF%BC%9A $dispatch %E5%92%8C,$broadcast %EF%BC%8C%E5%89%8D%E8%80%85%E7%94%A8%E4%BA%8E%E5%90%91%E4%B8%8A%E7%BA%A7%E6%B4%BE%E5%8F%91%E4%BA%8B%E4%BB%B6%EF%BC%8C%E5%8F%AA%E8%A6%81%E6%98%AF%E5%AE%83%E7%9A%84%E7%88%B6%E7%BA%A7%EF%BC%88%E4%B8%80%E7%BA%A7%E6%88%96%E5%A4%9A%E7%BA%A7%E4%BB%A5%E4%B8%8A%EF%BC%89%EF%BC%8C%E9%83%BD%E5%8F%AF%E4%BB%A5%E5%9C%A8%E7%BB%84%E4%BB%B6%E5%86%85%E9%80%9A%E8%BF%87 $on %EF%BC%88%E6%88%96 events%EF%BC%8C2.x %E5%B7%B2%E5%BA%9F%E5%BC%83%EF%BC%89%E7%9B%91%E5%90%AC%E5%88%B0%EF%BC%8C%E5%90%8E%E8%80%85%E7%9B%B8%E5%8F%8D%EF%BC%8C%E6%98%AF%E7%94%B1%E4%B8%8A%E7%BA%A7%E5%90%91%E4%B8%8B%E7%BA%A7%E5%B9%BF%E6%92%AD%E4%BA%8B%E4%BB%B6%E7%9A%84%E3%80%82)

[(64条消息) vue中 关于$emit的用法\_榴莲不好吃的博客-CSDN博客](https://blog.csdn.net/sllailcp/article/details/78595077)

大致有了个方向应该就是用create-react-app来搭建一个react组件库，并安装上一些配置

5/16

[使用ESLint+Prettier来统一前端代码风格 - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/38267286)

[史上最全Preitter指南(含option中文解释) - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/356654434)

[我理解的前端代码规范指南 - Prettier 篇 - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/608476939)

简单来说 eslint做的是代码语法的校验，而prettier用来规范代码风格。

今天开始用create-react-app搭建组件库，并配备了prettier以及eslintric。搭建完该组件库后，就开始研究静态按钮的编写，根据varlet-ui下的button的文件架构暂时决定了需要生成button.tsx存放button的react代码,button.css存放button的样式、prop.ts用来规定button的接口以及index.tsx存放Button组件的对外输出。接着就开始

下午5-6点听李老师分享ppt xml格式

[从0到1，搭建一个体系完善的前端React组件库 - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/171686818?utm_campaign=shareopn&utm_medium=social&utm_oi=1270999526832914432&utm_psn=1641749199984357376&utm_source=wechat_session)

5/17

下午算是完成了静态按钮，然后试着在npm发布了。（myapp文件夹下）

<https://www.npmjs.com/package/@flemingpau/myapp>

但是发现不对之前学过的webpack完全没用到，应该是要用到的，所以又搜了搜才发现原来vue对应的是react架构，而vite对应的是webpack，所以不应该直接用create-react-app于是就照着下面的网页重新构建了一次，这次重新搭建了一个组件库用的是webpack的方式。（mystore文件夹下）

[从零搭一个极简版webpack+React工程（一） - 掘金 (juejin.cn)](https://juejin.cn/post/7076438081225785375)

5/18

下午终于搭建完了一个比较简单的组件库webpack+react+ts了，接下来就是组件文档生成了，看到别人都是用的docz来生成，我也安装了试试，发现装不上，

应该是版本的问题，就问了袁哥发现貌似不需要安装这种插件，然后得知了使用nvm可以切换node的版本，还有主要是学习结构设计，webpack和vue的差别。于是就自己写了一点文档表示



5/19

看着varlet-cli的代码尝试拆分修改自己的webpack，一整天下来没搞明白

额外活动：中午外出庆祝，下午听李老师分享c++

5/22

感觉这样不是办法，就开始仔细查看varlet-cli的代码

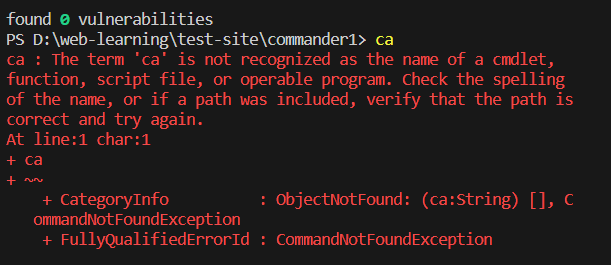
突然发现给的架构是要用pnpm 就重新安装了一次流程后面才发现其实删除node\_module和lock文件再用相应的pnpm来安装就可以了

看了两天也尝试写了写但是概念有点乱，写的十分混乱

Cli和组件库都不知道要干嘛，中间试着放弃去直接研究怎么写按钮

5/24

尝试给我的代码加上commander但是发现npm link后并没有出现全局命令



[【前端工程化】从入门到精通，100行代码构建你的前端CLI脚手架之路 - 掘金 (juejin.cn)](https://juejin.cn/post/7236021829000446011)

5/25

npm link全局弄到了直接把prefix改成nodejs下，之前看网上说nodejs下弄个node\_global我跟着做结果被误导了，别人的案例能正常跑了，我照着改一下

然后重新开始，之前的代码太乱了，重新开始整理代码

写了用cli如何从模板生成代码

5/26

下午继续听李老师讲c++

给cli成功用上了commander

5/29

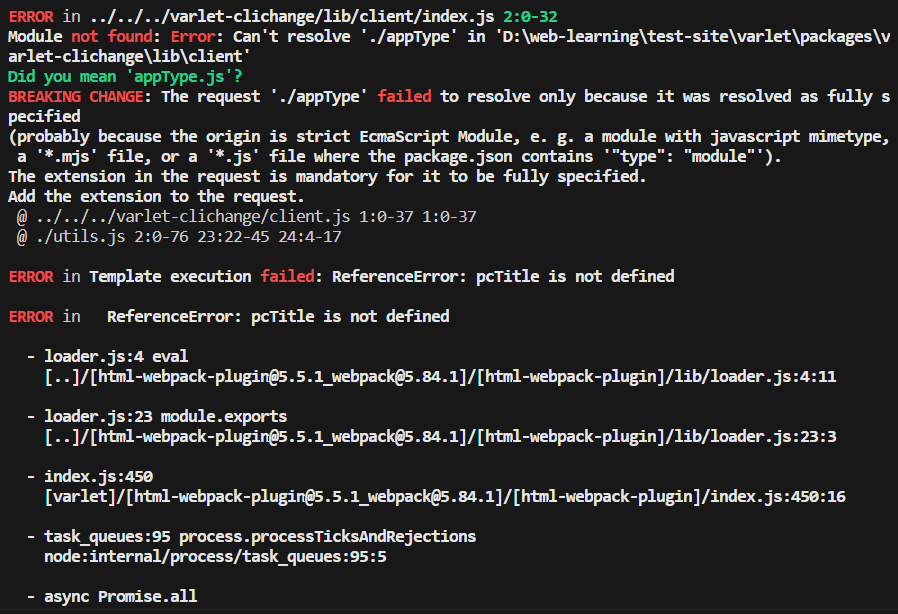
以为cli搞定了，尝试去搞组件库模板的完善，生成组件库。在试着修改webpack来拆分，拆不懂，放弃试着去写动态按钮，发现一个新的库叫proptypes，试着用

5/30

试着直接在varlet上修改成react和webpack的形式，越改越乱，感觉脑子秀逗了。

5/31

重新在varlet上修改，这次只改成用webpack+vue的形式能跑起来，再改成用webpack+react，改是改了，跑不起来



6/1

把cli的dev给补上了。然后给模块加上了monorepo

6/2

开始补上cli的production功能。写完的时候运行正常，我把模板里没用的文件删了，保留其目录格式，一运行发现报错，没有该全局命令。应该是对commander不小心由什么修改就重新link了一次，这次却运行不起来了，我检查了一遍，全局生成的位置，检查了两小时，才发现空文件夹被tsc编译没了导致目录对不上，运行不了

小问题：忘记了不可以给对象名用-

6/3

加上了buildComponent功能

6/5

早会-写报告，整理文档和代码

6/6

早会-写报告，整理文档和代码

6/7

写报告，整理文档

6/8

早会-写报告，整理文档

6/9

早会-实训结束-写报告，整理文档