模板内的表达式常用于简单运算，当过长或逻辑复杂时，使用计算属性

所有的计算属性都以函数的形式写在Vue实例内的computed选项内，最终返回计算后的结果

Computed内$refs获取貌似有问题（运行时间太早DOM还没加载完成），methods没问题

<input type="text" @input = "handleInput" placehonder="输入。。。">

<p>输入的内容是：{{message}}</p>

methods:{

handleInput : function(e){

this.message = e.target.value;

}

}

自己总结：事件函数带有可获取事件源的e

v-model 适用于表单元素

<input>、<textarea> 及 <select> 元素上创建双向数据绑定

v-model 会忽略所有表单元素的 value、checked、selected 特性的初始值而总是将 Vue 实例的数据作为数据来源。你应该通过 JavaScript 在组件的 data 选项中声明初始值。

v-model 在内部使用不同的属性为不同的输入元素并抛出不同的事件：

text 和 textarea 元素使用 value 属性和 input 事件；

checkbox 和 radio 使用 checked 属性和 change 事件；

select 字段将 value 作为 prop 并将 change 作为事件。

表单元素v-model应该配合value使用，否则会有异常

*<!--组合使用实现 互斥 效果，需使用 v-model和value 配合-->*

<input type="radio" v-model="picked1" value = "html" id="html">

<label for="html">HTML </label>

<br>

<input type="radio" v-model = "picked1" value="js" id="js">

<label for="js">Javascript</label>

<br>

<input type="radio" v-model = "picked1" value="css" id="css">

<label for="css">CSS </label>

<br>

<p>选择的项是:{{picked1}}</p>

*<!-- 多个复选框，绑定到同一个数组 -->*

<input type="checkbox" id="bazi" value="bazi" v-model="checkedNames">

<label for="bazi">bazi</label>

<input type="checkbox" id="shiyi" value="shiyi" v-model="checkedNames">

<label for="shiyi">shiyi</label>

<input type="checkbox" id="luxin" value="luxin" v-model="checkedNames">

<label for="luxin">luxin</label>

<br>

<span>Checked name:{{checkedNames}}</span>

picked1和checkedNames 输出的值取决于在data中申明的类型，如果是字符串则：redio：字符串 互斥并输出选中值 checkbox 会变成布尔值 true、false互斥

数组：redio依旧互斥并只输出选中值 checkbox 输出所有选中值

选择列表：

优先读取value值，其次text文本

v-model 配套的绑定值 true-value false-value 得与一个true/false值一起使用

对于单选按钮，复选框及选择框的选项，v-model 绑定的值通常是静态字符串 (对于复选框也可以是布尔值)：

*<!-- 当选中时，`picked` 为字符串 "a" -->*

<input type="radio" v-model="picked" value="a">

*<!-- `toggle` 为 true 或 false -->*

<input type="checkbox" v-model="toggle">

*<!-- 当选中第一个选项时，`selected` 为字符串 "abc" -->*

<select v-model="selected">

<option value="abc">ABC</option>

</select>

v-model.lazy 失焦或者回车起效 input事件变为change事件

组件

注意项：①必须在实例化Vue之前注册组件，不然报以下错误：

Unknown custom element: <button-counter> - did you register the component correctly? For recursive components, make sure to provide the "name" option.

注意项：②直接引入Vue.js的情况下，在编写组件的template时必须要有元素包裹

Vue.component('button-counter',{

data:function(){

return {

count:0

}

},

template:'<button v-on:click="count++">{{count}}</button>'

template:'{{count}}'*//缺少包裹元素*

})

**！！！**Vue组件的模板在某些情况下会受到HTML的限制，比如<table>内规定只允许时<tr><td><th>这些表格元素，所以在<table>内直接使用组建是无效的。

这种情况下，可以使用特殊的 is 属性来挂载组件—经测试可直接使用

**[data 必须是一个函数](https://cn.vuejs.org/v2/guide/components.html" \l "data-必须是一个函数" \o "data 必须是一个函数)**

当我们定义这个 <button-counter> 组件时，你可能会发现它的 data 并不是像这样直接提供一个对象：

data: {

count: 0

}

取而代之的是，**一个组件的 data 选项必须是一个函数**，因此每个实例可以维护一份被返回对象的独立的拷贝：[在子组件里return的props也是独立的不影响父组件内的值]

data: function () {

return {

count: 0

}

}

var data={

counter :0

}

Vue.component('button-counter',{

data:function(){

return data

},

template:'<button v-on:click="counter++">{{counter}}</button>'

})

使用多个button-counter 组件时，点击事件将会让所有该组组件的counter同时+1，而不是单独处理每个组件

对于 components 对象中的每个属性来说，其属性名就是自定义元素的名字，其属性值就是这个组件的选项对象。

注意**局部注册的组件在其子组件中*不可用***。例如，如果你希望 ComponentA 在 ComponentB 中可用，则你需要这样写：

var ComponentA = { /\* ... \*/ }

var ComponentB = {

components: {

'component-a': ComponentA

},

// ...

}

或者如果你通过 Babel 和 webpack 使用 ES2015 模块，那么代码看起来更像：

import ComponentA from './ComponentA.vue'

export default {

components: {

ComponentA

},

// ...

}

注意在 ES2015+ 中，在对象中放一个类似 ComponentA 的变量名其实是 ComponentA: ComponentA 的缩写，即这个变量名同时是：

* 用在模板中的自定义元素的名称
* 包含了这个组件选项的变量名
* **[在模块系统中局部注册](https://cn.vuejs.org/v2/guide/components-registration.html" \l "在模块系统中局部注册" \o "在模块系统中局部注册)**
* 如果你还在阅读，说明你使用了诸如 Babel 和 webpack 的模块系统。在这些情况下，我们推荐创建一个 components 目录，并将每个组件放置在其各自的文件中。
* 然后你需要在局部注册之前导入每个你想使用的组件。例如，在一个假设的 ComponentB.js 或 ComponentB.vue 文件中：
* import ComponentA from './ComponentA'
* import ComponentC from './ComponentC'
* export default {
* components: {
* ComponentA,
* ComponentC
* },
* // ...
* }
* 现在 ComponentA 和 ComponentC 都可以在 ComponentB 的模板中使用了。

## [通过 Prop 向子组件传递数据](https://cn.vuejs.org/v2/guide/components.html" \l "通过-Prop-向子组件传递数据" \o "通过 Prop 向子组件传递数据)

一个组件默认可以拥有任意数量的 prop，任何值都可以传递给任何 prop。

当你使用 DOM 中的模板时，camelCase (驼峰命名法) 的 prop 名需要使用其等价的 kebab-case (短横线分隔命名) 命名 使用字符串模板，那么这个限制就不存在了

在组件中，使用选项props来声明需要从父级接受的数据，props的值可以是两种，① 字符串数组 ②对象

字符串数组中

通常你希望每个 prop 都有指定的值类型。这时，你可以以对象形式列出 prop，这些属性的名称和值分别是 prop 各自的名称和类型：

props: {

title: String,

likes: Number,

isPublished: Boolean,

commentIds: Array,

author: Object

}

**！！静态数据直传而不是用bind传递的仅为字符串 建议均使用bind**

*// 对比静态值直传和bind传递效果*

<my-component4 ai-bi="[1,2,3]"></my-component4>//7

<my-component4 :ai-bi="[1,2,3]"></my-component4>//3

Vue.component('my-component4',{

props:['aiBi'],

template:'<div>{{aiBi.length}}</div>'

})

这里有两种常见的试图改变一个 prop 的情形：

1. **这个 prop 用来传递一个初始值；这个子组件接下来希望将其作为一个本地的 prop 数据来使用。**在这种情况下，最好定义一个本地的 data 属性并将这个 prop 用作其初始值：
2. props: ['initialCounter'],
3. data: function () {
4. return {
5. counter: this.initialCounter
6. }

}

1. **这个 prop 以一种原始的值传入且需要进行转换。**在这种情况下，最好使用这个 prop 的值来定义一个计算属性：
2. props: ['size'],
3. computed: {
4. normalizedSize: function () {
5. return this.size.trim().toLowerCase()
6. }

}

不同于组件和 prop，事件名不会被用作一个 JavaScript 变量名或属性名，所以就没有理由使用 camelCase 或 PascalCase 了。并且 v-on 事件监听器在 DOM 模板中会被自动转换为全小写 (因为 HTML 是大小写不敏感的)，所以 v-on:myEvent 将会变成 v-on:myevent——导致 myEvent 不可能被监听到。

因此，我们推荐你**始终使用 kebab-case 的事件名**。

子组件向父组件传值

子组件用$emit()来触发事件，父组件用$on()来监听子组件的事件

v-model在组件上的使用可以分解为父组件监听input事件，子组件触发input事件并传值

非父子通信：①兄弟组件②跨多级组件

实现方法：1.中央事件总线（空Vue实例 var event Bus= new Vue（））传递并接收事件

let Hub = new Vue(); //创建事件中心  
组件1触发：  
methods: { eve() { Hub.$emit('change','hehe'); //Hub触发事件 } }  
组件2接收:  
created() { Hub.$on('change', () => { //Hub接收事件 this.msg = 'hehe'; }); }

2.父链：在子组件中，使用this.$parent可以直接访问该组件的父实例或者组件，父组件也可以通过this.$children访问它所有的子组件，而且可以递归向上或者向下无限访问，直至根实例或者最内层的组件

3.子组件索引

注：props传递数据、events出发时间和slot内容分发构成Vue组建的3个API来源

Slot：内容分发 只能用于template内

<children>

{{message}} ==>slot，绑定父组件数据，而不是组建<children>的数据

</children>

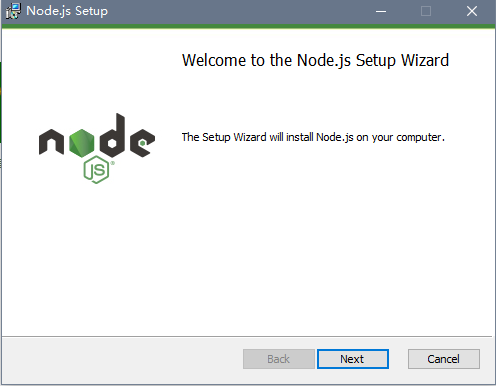
编译作用域：父组件模板的内容是在父组件作用域内编译，子组件模板的内容是在子组件作用域内编译

Slot分发的内容，作用域是在父组件上？？？

Vue-cli

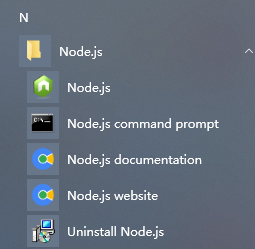
Vue-cli3.X

下载脚手架Vue-cli的第一步：下载安装Node.js 百度Node.js官网 或者Node.js中文网均可 如无特殊要求 下载安装 LTS版（长期支持版，该版本已满足Vue CLI官网所需要求版本要求）[以Windows为例]



一路Next即可，一般无需更改默认安装设置

安装完毕在开始菜单内即可找到一个名为 Node.js 的文件夹如下图



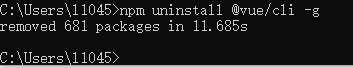
注：Node.js内可直接运行js基础代码（非DOM等相关代码）

而我们需要使用的是 Node.js command prompt

官方安装命令 npm install -g @vue/cli

注：@不能少，少了@安装的不是3.x版本而是1.x、2.x版本，已经安装了旧版本的需要卸载旧版本然后再安装新版本 旧版本卸载命令 npm uninstall vue-cli -g

vue-cli 3.x 卸载命令 npm uninstall @vue/cli -g (如果需要可使用)



国内下载速度比较慢

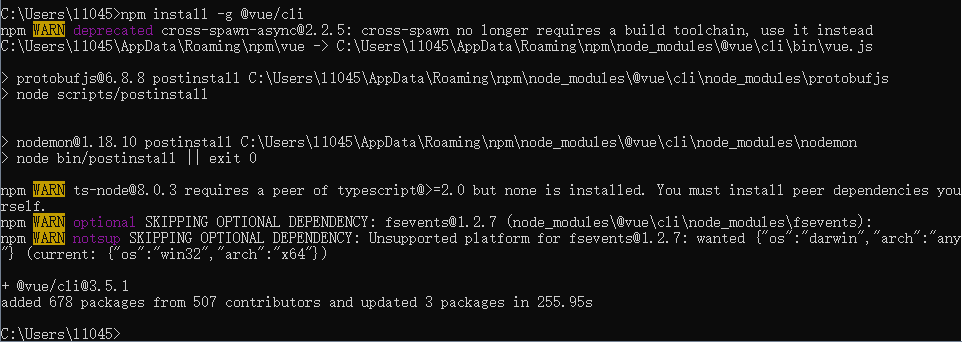
查看npm下载镜像的源地址，命令如下：

npm get registry

结果> https://registry.npmjs.org/

如果嫌弃下载慢可以切换为淘宝镜像，命令如下：

npm config set registry http://registry.npm.taobao.org/



接着就是创建项目，但是建议先考虑项目文件夹储存的位置并进入相关位置，以桌面为例

cd desktop

如果无需更改位置，请跳过；如果需返回上级目录 cd.. 即可，其他cmd命令可自行百度

创建项目的两种方式：

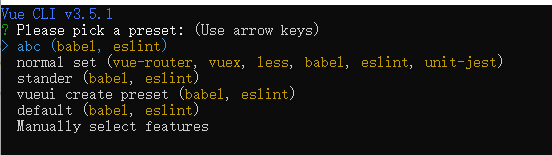
①直接命令行创建

创建项目命令： vue create 项目名 自己起个名字即可，但请注意不能有大写字母，否则报错：Warning: name can no longer contain capital letters

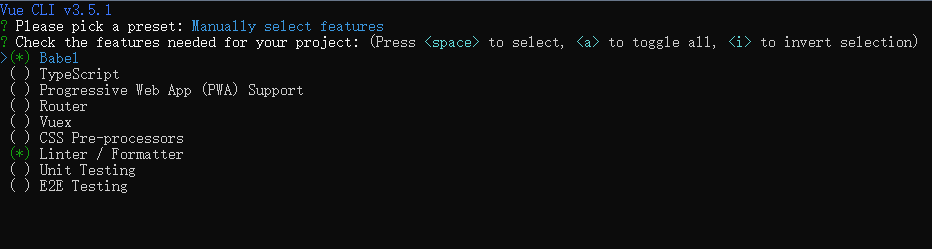
vue create learn √

接下来会问你选择那种配置（会显示所有保存过的配置，首次使用只会显示 default（babel,eslint） 以及 Manually select features）

可选自己之前创建过的或选择官方默认配置或者手动选择



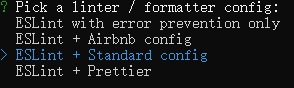
选择Manually select features 将出现以下内容



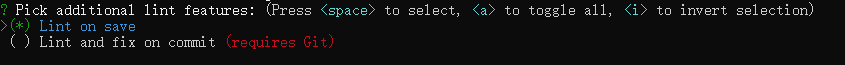
方向键 控制上下 空格选择 A 全选 I反转选择

选择好大体的功能/插件之后 回车 接着会让你选择细分的特性

比如 Linter/Formatter 下：



在保存时进行代码检测



Babel、PostCSS、ESLint等配置文件存放位置，选择单独保存在各自的配置文件中



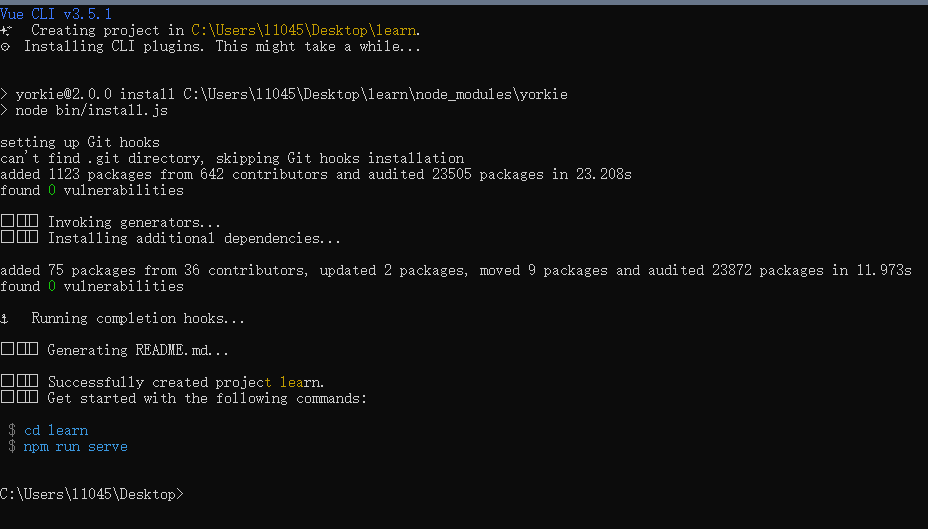
Pick a CSS pre-processor(css预处理语言)下 有SCSS/SASS、LASS、Stylus

还有TS、PWA、Roter（vue 路由管理；SPA必带）、Vuex（vue状态管理）、测试（Unit Testing、E2E Testing）等等，均可按需选择

最后会询问你 是否保存为未来项目的预设配置 如果保存需要输入名称

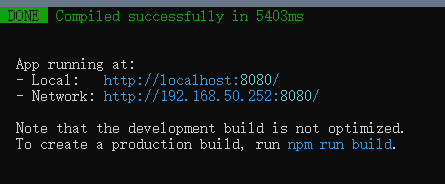


等待项目创建...



项目创建好后按提示进入项目，并启动项目

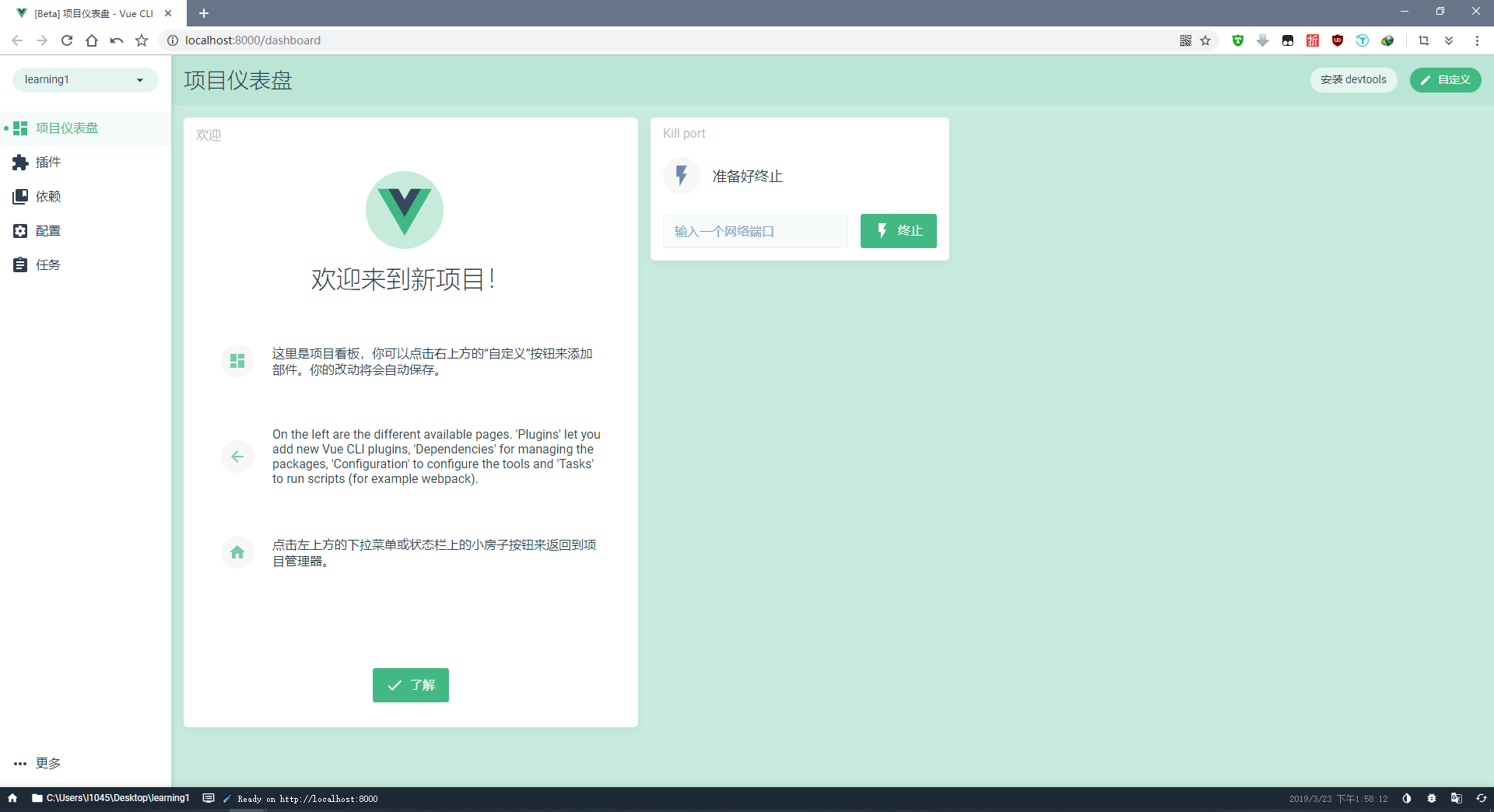
cd learn 进入后 npm run serve



这两个地址均可访问，第二个地址也并不能对外访问，应该是没做好端口映射，那这个地址用来干嘛的？

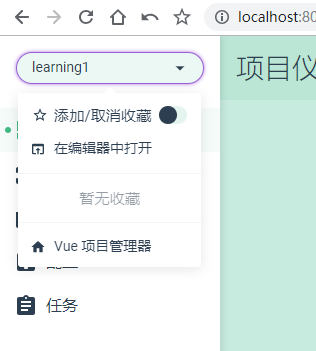
通过GUI创建

在下载好vue-cli3.x后，进入项目存储位置后 输入 vue ui，会在浏览器内打开vue项目仪表盘（非初次使用界面）

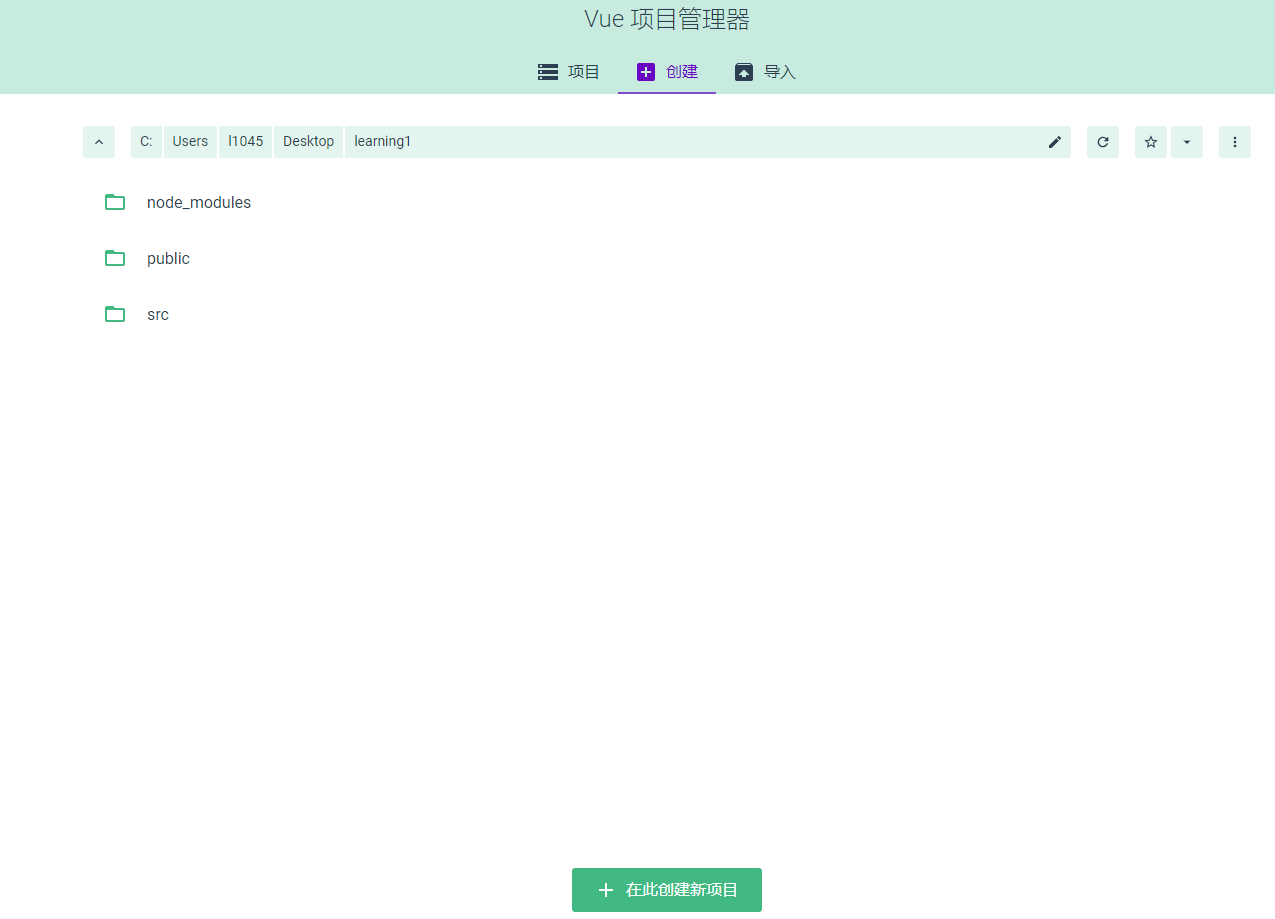


在使用过GUI创建过项目后，再次打开就会直接进入旧项目管理界面，如上

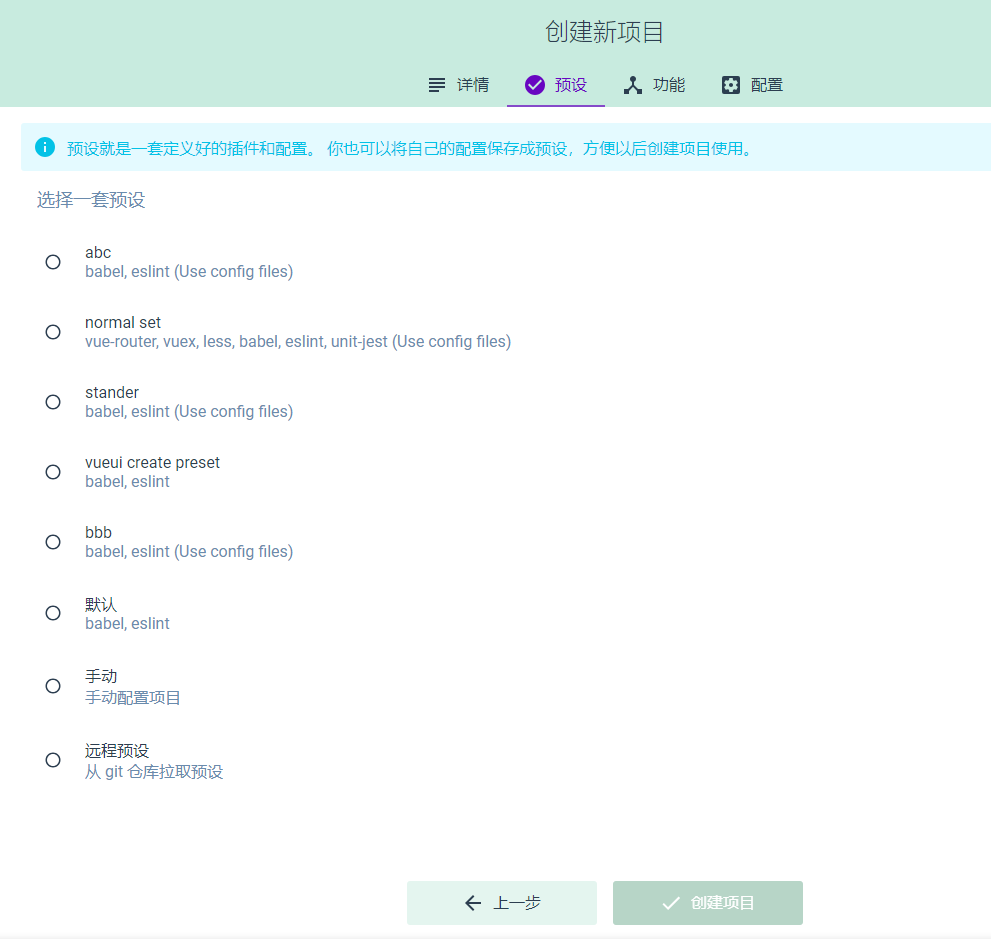
如果需要再新建一个项目，可在 左上角下拉菜单内选择 ”Vue项目管理器“返回首页





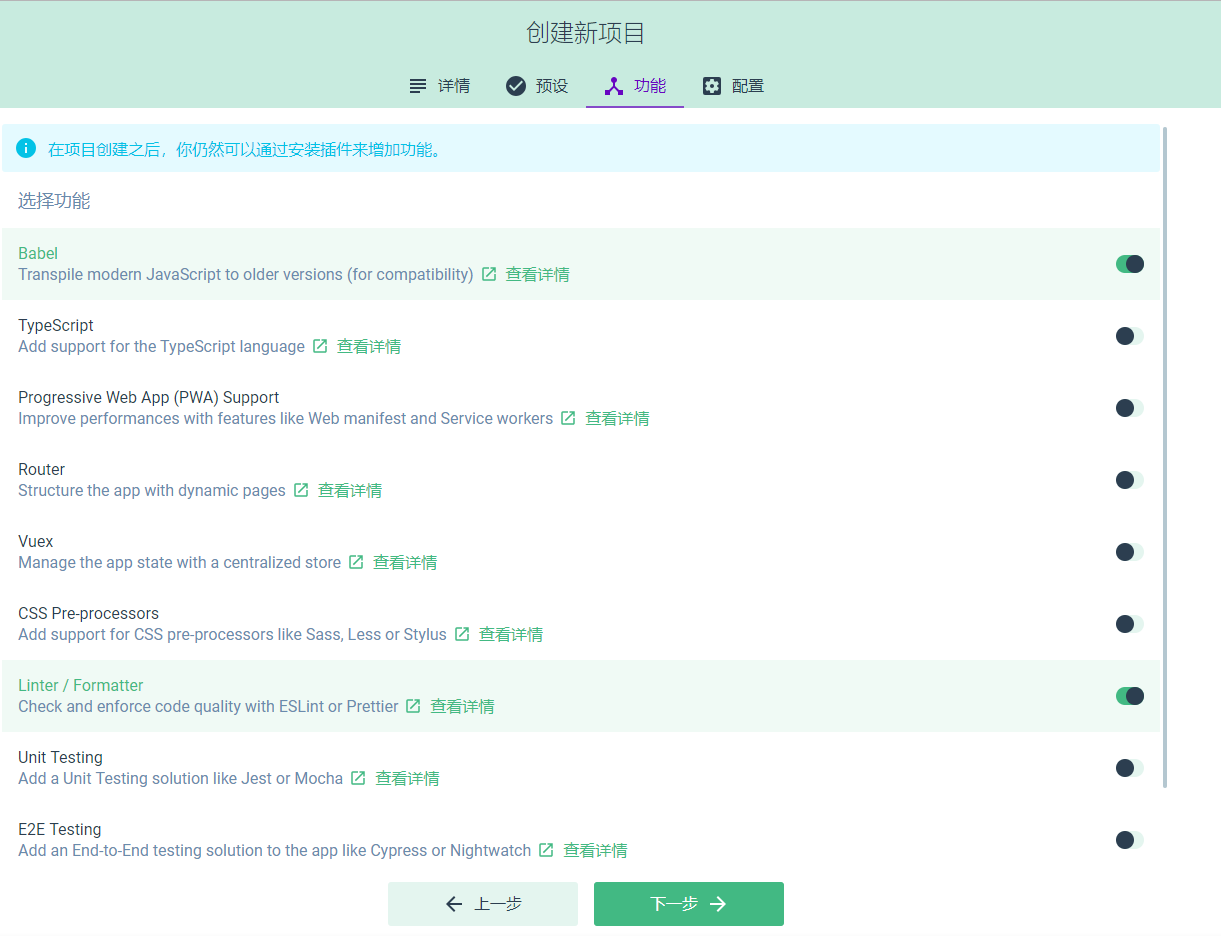


创建——》在此创建新项目——》输入项目文件夹名（没有git建议关闭）——》下一步



可选自己创建过的配置或者官方配置或者手动配置

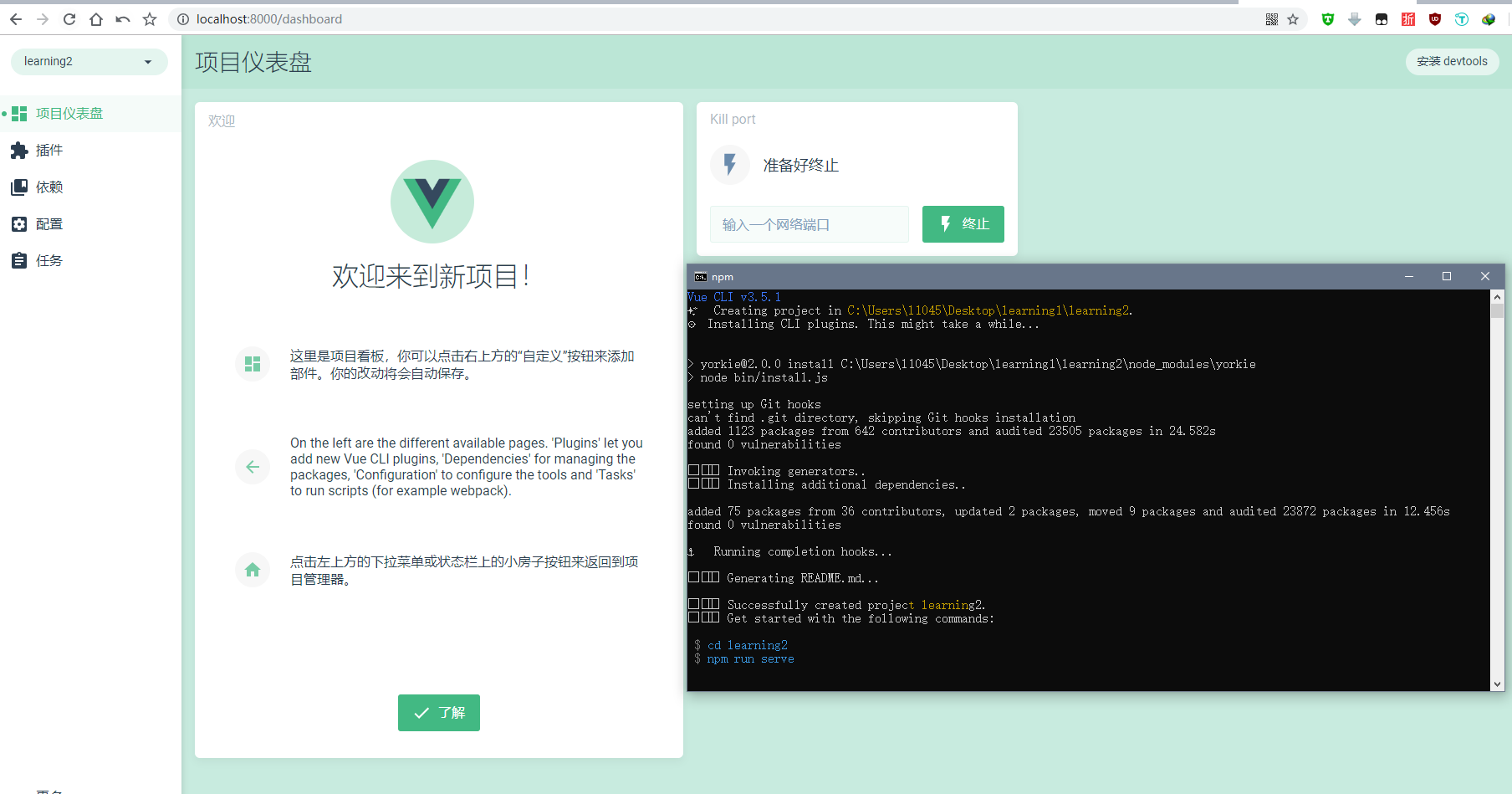
手动配置如下图所示， 按需选择，相关介绍之前提到过了



注：有下拉框的得先选择否则不能 下一步

最后点击创建并等待

最终和CMD截图如下



如果想在浏览器内启动的话 左侧菜单栏——》任务——》任务面板——》serve——》运行——》右侧 启动app (从左往右点击红色即可)



也可命令行启动，命令与之前所述相同

至此结束