### 一．vue

Day1 ················································1

Day2 ················································1

Day3 ················································1

Day4 ················································1

Day5 ················································1

Day6 ················································1

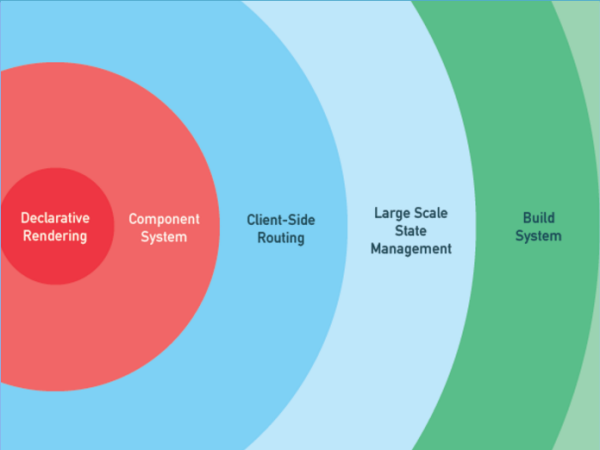
**Day1**

场景1：  
如果你的技术团队技术选型比较保守，没有新技术的使用场景，比如让你去维护一个管理后台。管理后台，日常就是提交各种表单了，这部分现有的方案，比如form表单提交或者jquery收集信息ajax提交。这时候你可以把vue 当成一个js库来使用，就用来收集form表单，和表单验证。

场景2：  
在场景1中，你尝到了甜头。心中暗爽，还可以这么玩嘞。独乐乐不如众乐乐和大家分享之后，团队开始接受使用vue，小规模推广起来。打怪升级该遇小boss啦，领导说，小伙后台做的越来越有效率了，来承担些常规业务开发，来正规军编制和其他小伙伴一起做新闻列表和新闻详情页吧。  
在这个项目中，你跟大胆一点把 整个页面的dom 用Vue 来管理，你发现jquery 没什么用了，列表用v-for来循环，把评论抽成小组件了。 评论交互比较复杂，但是你的关注点把原来jquery dom操作变成了关注数据的变化，用数据驱动DOM的变化

场景3:  
经历了场景1 场景2，越来越受大家信赖，领导又找你了。你看新闻项目你做的不错，移动站也得重构了，你做个移动端m站吧，正好微信和App分享出去用到。这时候，你需要在做移动端webapp了。 于是你由去了解 webpack vue-router，你发现前端 可以控制路由了。webpack，可以用于前端开发的工程构建。

场景4:  
场景3之后你在技术团队大放异彩，公司越来越器重你。年末领导又来找你了，小伙砸，想不想拿年终奖啊，想的话给我做个新闻直播间吧。需求特别简单：就是滚动播报新闻，用户实时参与评论。有了场景3和之前的经验，发现稍微有点力不从心了，你和后端的接口沟通上越来越频繁，新闻直播间需要大量的数据在组件中共享数据，后来你发现了，vuex 处理数据在组件之间的流动得心应手。场景5:  
年终奖，拿到手了，过年回来升职加薪。带了20人的前端团队，你的精力开始在配合公司其他部门做用户数据增长了。发现场景2中你犯了个错误，虽然整个页面用Vue 管理 开发起来很方便，但是页面白屏时间长，而且类似这样的底层页对seo都不好。开始考虑使用 vue2.0的SSR。为了保障团队高质量的输出，你开始研究如何给 vm写单测...

场景1-5 从最初的只因多看你一眼而用了前端js库，一直到最后的大型项目决方案。能否给你说明白vue 是 The ProgressiveJavaScript Framework  


vue.js

day1:

vue的概述

vue的常见指令和插值表达式

vue的组件的基本概念

day2:

vue的组件高级用法

vue不同的组件之间如何进行数据通信

day3:

vue的路由模块使用

vue的过渡效果

vue与远程服务器端的通信

WebStorm提供了一个模板的功能，常见一个模板：

①编写模板

②拷贝模板当中的所有的内容

③点击webstorm左上角的file--》new--》edit file templates

④点击弹窗中的左上角的加号，指定模板的名字（vue）和结尾的名字（html）

⑤保存就可以

零、框架

①什么是框架？

框架指的是一套非常优秀可被反复使用的代码，帮助开发者解决一些复杂的问题。

②框架有啥好处？

提高了代码的复用率

降低模块之间的耦合度

提高开发速度

提高代码质量

一、Vue的概述

Vue的开发模式 和 之前接触的jQuery、原生JSDOM操作是不同的，之前要想修改试图，首先找元素；在使用Vue时，专心把精力放在修改数据。

DOM驱动 ---》 数据驱动。

what？

Vue的开发者是尤雨溪，是受到angular的启发，是一个实现ui层的渐进式的js框架

版本：V2.4.0

注意：

①并不是所有的ui类的库或者框架 都提供了丰富的样式类

②渐进式：step by step

where？

饿了么 滴滴 阿里巴巴

gitlab。。。

VueJS可以用在很多地方，从简单的表单验证，到复杂的大型企业应用程序，Vue都可以胜任；核心库是实现UI层的。

why？

①非常好的中文文档

②学习曲线比较缓和（容易上手）

③速度快

④体积小

⑤基于组件化（web component）的开发方式

⑥代码的可读性、可维护性比较好

how？

搭建环境有两种方式：

方式1：CLI（command line interface 命令行窗口）

#安装nodejs

https://nodejs.org/en/

#全局安装 vue-cli

npm install --global vue-cli

#创建一个基于 webpack 模板的新项目

vue init webpack my-project

#安装依赖，走你

cd my-project

npm install

npm run dev

方式2：直接引入对应的js文件

二、Vue的基础使用

1、第一个例子

通过new的方式来得到了Vue的实例，在new通过一个对象来对Vue的实例配置（el，data）；

接触了双花括号(mustache/interpolation)的语法:可以读取变量的值 显示在调用的元素的innerHTML中。

练习时间：

新建一个文件，通过vue的方式 ，在html中执行

一个三目运算、一个逻辑运算；将运算的结果显示不同的h1标签中。

2、双花括号

语法： <any>{{表达式}}</any>

作用：执行表达式，将表达式的结果 输出到当前调用元素的innerHTML中

三、Vue的常用指令

循环、选择判断、显示隐藏。。。

1、循环指令

语法1：

<any v-for=" tmp in 集合"></any>

根据in关键词后的集合，去循环创建多个标签，

举例：

myList:[100,200,300]

<a v-for="tmp in myList">{{tmp}}</a>

语法2：

<any v-for="(value,key) in 集合"></any>

练习：

去新建一个文件，通过Vue将一个对象数组 显示在table(thead和tbody)中。

对象数组有5个对象，

每个对象指定3个属性：name/sex/age.

基本步骤

①定义数据

[{},{}]

②在视图中创建table

③将数据通过v-for和双花括号将值显示出来

2、选择指令

<any v-if="表达式"></any>

v-if指令 ： 根据表达式执行的结果的真假 来选择是否要挂载到DOM

v-else-if

v-else

练习：

通过Vue，渲染一个h1,ul.

在刚开始时，h1：正在加载数据 ,ul显示一个空的列表在vue实例创建完毕1s之后，h1:从DOM中移除掉，同时给数组初始化5条数据 显示在ul中。

data需要用到的数据：

isDataReady:false

myList:[]

①搞定数据

②显示在视图中

③指定定时器

3、Vue事件绑定

<any v-on:eventName="处理函数名字"></any>

通过v-on去给指定的事件绑定一个处理函数

举个例子：

<button v-on:click="handleClick">clickme</button>

当触发click事件，就会调用handleClick方法。

注意事项：

方法在定义时，要放在Vue实例的methods属性

new Vue({

methods:{

handleClick:function(){

//做一些操作

}

}

})

事件绑定支持支持一种简写形式：

<button v-on:click="handleClick">clickme</button>

//通过@后边跟上事件的名字，和v-on:是一样的效果

<button @click="handleClick">clickme</button>

Vue.js 为 v-on 提供了 事件修饰符。通过由点(.)表示的指令后缀来调用修饰符。

v-on:submit.prevent="handleSubmit"

4、Vue其它常用指令

v-bind

将变量中的值 通过 v-bind 绑定到元素指定的属性

img v-bind:src="myImage"

a v-bind:href="myLink"

v-bind:style='{backgroundColor:myBGColor}'

v-bind:class='{myRed:isRed}'

v-bind:disabled="!isValid"

v-bind也支持简写

a v-bind:href="myLink"

===》

a :href="myLink"

v-show

v-show="表达式" 根据表达式执行的结果的真假 来切换display显示还是隐藏

v-text

v-text='变量' 根据变量更新对应的文本内容

v-html

v-html="变量" 根据变量更新对应的innerHTML

...

5、Vue中的v-model

双向数据绑定：

方向1：data --> view

之前所接触的循环、判断、显示隐藏等指令，主要的功能是将数据 绑定到视图；

方式：①双花括号 ②常见的指令

方向2：view --》 data

将表单控件中 用户操作的 结果 绑定到 数据

方式：v-model

v-model指令本身就是一个双向数据绑定的指令：既可以将数据绑定到视图，又可以将视图中的结果绑定到数据。

修饰符：

.lazy - 取代 input 监听 change 事件（在输入完之后，失去焦点 再去更新）

.number - 输入字符串转为数字

.trim - 输入首尾空格过滤

每日一练：

通过Vue来实现一个分页的效果。

提供一个数组：

pageNumbers=[1,2,3,4,5]

将数组显示在一个列表中，默认第0个被选中，点击列表项时候，让该列表项高亮显示。

v-for

v-bind

v-on

**Day2**

每日一练：

通过Vue来实现一个分页的效果。

提供一个数组：

pageNumbers=[1,2,3,4,5]

将数组显示在一个列表中，默认第0个被选中，点击列表项时候，让该列表项高亮显示。

v-for

v-bind

v-on

day2学习安排：

自定义指令

自定义过滤器

组件的创建和使用

组件的高级用法

一、自定义指令的创建和使用

Vue自带的指令很多，v-for/v-if/v-else/v-else-if/v-model/v-bind/v-on/v-show/v-html/v-text...

但是这些指令都是比较偏向于工具化，有些时候在实现具体的业务逻辑的时候，发现不够用，如何来自定义指令.

1、自定义指令

①创建

new Vue({

directives:{

change:{

bind:function(){},

update:function(){},

unbind:function(){}

}

}

})

在自定义指令时，在指令对应的配置对象中有3个处理函数指令对应的操作

bind

指令在绑定到元素要执行的操作

update

如果在调用指令时候，传了参数，当参数变化时候，就会执行update所指定的方法

unbind

解绑要执行的操作

②使用自定义指令

directives:{

hello:{

bind:function(){},

update:function(){},

unbind:function(){}

}

}

v-hello

注意事项：

建议在给指令的命名采用小驼峰式的命名方式，比如changeBackgroundColor,在使用的时候，采用烤串式写法 v-change-background-color

练习：实现一个自定义指令v-change-background-color,

changeBackgroundColor

在调用指令时，可以去传递参数，根据参数的值 修改调用该指令的元素的背景色。

二、自定义过滤器的创建和使用

过滤器：生活中有很多例子，净水器 空气净化器 。

过滤器的作用：实现数据的筛选、过滤、格式化。

vue1.\*版本是有内置的过滤器，但是在vue2.\*所有的版本都已经没有自带的过滤器了。

1、过滤器创建

过滤器的本质 是一个有参数 有返回值的方法

new Vue({

filters:{

myCurrency:function(myInput){

return 处理后的数据

}

}

})

2、过滤器使用

语法：

<any>{{表达式 | 过滤器}}</any>

举个例子：

<h1>{{price | myCurrency}}</h1>

3、过滤器高级用法

在使用过滤器的时候，还可以指定参数，来告诉过滤器按照参数进行数据的过滤。

①如何给过滤器传参？

<h1>{{price | myCurrency('￥',true)}}</h1>

②如何在过滤器中接收到？

new Vue({

filters:{

//myInput是通过管道传来的数据

//arg1在调用过滤器时在圆括号中第一个参数

//arg2在调用过滤器时在圆括号中第二个参数

myCurrency:function(myInput,arg1,arg2){

return 处理后的数据

} }})

练习：

实现一个过滤器myTextTransform,功能是实现将一个字符串转换为大写或者小写，取决于传递给过滤器的参数，如果是true转换为大写 否则小写。

myInput.toUpperCase()

myInput.toLowerCase()

三、组件

w3c为什么有组件概念？

现在的前端领域，对于代码的质量和数量有着非常高的要求，特别容易出问题。

通过组件的方式来完成代码的管理和编写。

组件：是一个可以被反复使用的 带有指定功能的 视图

组件化编程 一切都是组件component，把可以被反复使用的任何的视图 都可以封装成组件。

类似玩积木。

组件化的好处（封装）：

①提高了代码的复用率

②提高开发速度

③降低测试难度

④降低整个应用的耦合度

。。。

1、组件的创建

Vue.component('my-component',{

template:'<h1>it is a component</h1>'

})

2、组件的使用

组件的使用就像是使用普通的html标签一样

<my-component> </my-component>

建议：组件的命名和使用 建议来使用烤串式命名规则。

（如果组件的名字写成驼峰式 myComponent,使用依然还得是烤串式）

注意：

①如果一个组件要渲染多个元素，将多个元素放到一个顶层标签；否则会报错。

3、复合组件

组件可以像html一样来使用，复合组件并不是一个新的概念，就是一个组件，只不过该组件中又调用了其它的组件。

一个完整的Vue的项目，可以比作是一颗组件树，组件树的根 就是 根组件（my-article）....

组件树可以帮助我们降低业务的复杂度，避免出现一些错误，提高开发速度。

4、组件的生命周期

分为4个阶段：

create/mount/update/destroy

每一个阶段都对应着有自己的处理函数

create: beforeCreate created

初始化

mount: beforeMount mounted

和挂载相关的处理

update: beforeUpdate updated

根据要更新的数据 做逻辑判断

destroy:beforeDestroy destroyed

清理工作

在Vue的实例或者在组件，都有哪些属性？

el

data

methods

directives

filters

watch

练习：

生命周期。

创建一个组件，该组件渲染一个h1的标签。

要求：在组件挂载完毕之后，启动定时器，修改h1标签的透明度的值 0~1

v-bind:style='{backgroundColor:myBGColor}'

每日一练：

创建一个组件，my-game:

猜数字大小。

组件：一个input和一个p构成

当组件准备挂载的时候，初始化一个随机数，在input取输入的时候，如果输入的数字小了，在p显示：输入的太小了；如果输入的数字大了，在p显示：输入的太大了；否则就提示输入正确。

**Day3**

每日一练：

创建一个组件，my-game:

猜数字大小。

组件：一个input和一个p构成

当组件准备挂载的时候，初始化一个随机数，在input取输入的时候，如果输入的数字小了，在p显示：输入的太小了；如果输入的数字大了，在p显示：输入的太大了；否则就提示输入正确。

学习安排：

组件间通信

父-》子(props down)

子-》父 (events up)

兄弟组件

<a href='http://www.tedu.cn'>

一、组件间通信（父组件--》子组件）

步骤：

①父组件在调用子组件 传值

<child-component myValue="123"> </child-component>

②在子组件中 获取父组件传来的值

Vue.component('child-component',{

props:['myValue'],

template:''

})

练习：登录窗口

创建4个组件，分别是my-label my-input my-button my-login(复合组件)

二、组件间通信（子组件传值给父组件）

通过事件的方式来完成数据的传输。

①在父组件中 定义一个方法，用来接收子组件所通过事件传来的值

methods:{

recvMsg:function(msg){

//参数msg就是子组件通过事件出来的数据

}

}

②绑定事件处理函数

事件一般情况 都是自定义事件

<child-component @myEvent="recvMsg">

</child-component>

③在子组件触发事件

this.$emit('myEvent',myPhone)

//触发一个叫做myEvent的事件，同时把第二个参数数据传递给事件对应的处理函数

总结：

在 Vue 中，父子组件的关系可以总结为 props down, events up。父组件通过 props 向下传递数据给子组件，子组件通过 events 给父组件发送消息。

练习：

在子组件中 放上一个input，点击按钮 把用户输入的内容发给父组件

练习：

有两个组件，分别是main-component,header-component.

main-component是由header-component和一个列表（有5条数据 [100,200,300,400,500]），

header-component是由一个h1的标签：'这是页头',有一个数据isUserLogin:true

在渲染main-component时候，读取header-component在挂载完毕之后通过事件传递来的数据（isUserLogin），根据该数据的真假来决定列表是否显示.

<child-component @myEvent="recvMsg">

</child-component>

this.$emit('myEvent',myPhone)

三、组件间通信（$parent $refs）

父组件要想获取子组件的数据：

①在调用子组件的时候，指定ref属性

<child-component ref="mySon"></child-component>

②根据指定的引用的名字 找到子组件的实例对象

this.$refs.mySon

子组件要想获取父组件的数据：

①直接读取

this.$parent

组件间通信综合练习：

（props down,events up）

有2个组件：chat-room,user-component

user-component是由label input button构成

chat-room是由两个user-component和一个列表构成

①在chat-room调用user-component指定label的名字

②在user-component，

点击按钮时，将当前用户输入的信息发送给chat-room组件，chat-room接收到数据显示在列表中

四、兄弟组件间通信（event）

借助于一个公共的Vue的实例对象，不同的组件可以通过该对象完成事件的绑定和触发

var bus = new Vue();

bus.$emit()

bus.$on()

熊大想要发消息给熊二，

接收方（熊二）：事件绑定

bus.$on('customEvent',function(msg){

//msg就是通过事件 传递来的数据

})

发送方（熊大）：触发事件

bus.$emit('customEvent',123);

练习：

在熊二中 加上一个button，

点击按钮时告诉熊大:'快跑！'

接收方：事件绑定

发送方：触发事件

总结：

父组件--》子组件

①通过属性

步骤1：

<son myName="michael" myPhone='123'></son>

<son :myName="userList[0]"></son>

步骤2：

Vue.component('son',{

props:['myName','myPhone']

})

②通过$parent

直接在子组件中通过this.$parent得到调用子组件的父组件

子组件--》父组件

①events up

步骤1：在父组件中 调用子组件的时候 绑定一个自定义事件 和 对应的处理函数

methods:{

recvMsg:function(msg){

//msg就是传递来的数据

}

}，

template:'

<son @customEvent="recvMsg"></son>

'

步骤2：在子组件中 把要发送的数据通过触发自定义事件传递给父组件

this.$emit('customEvent',123)

②$refs

reference 引用

步骤1：在调用子组件的时候 可以指定ref属性

<son ref='zhangsan'></son>

步骤2：通过$refs得到指定引用名称对应的组件实例

this.$refs.zhangsan

兄弟组件通信

步骤1：创建一个Vue的实例 作为事件绑定触发的公共的对象

var bus = new Vue();

步骤2：在接收方的组件 绑定 自定义的事件

bus.$on('customEvent',function(msg){

//msg是通过事件传递来的数据 (传递来的123)

});

步骤3：在发送方的组件 触发 自定义的事件

bus.$emit('customEvent',123);

每日一练：

创建2个组件，main-component,son-component

视图：

main-component 显示一个按钮

son-component 显示一个p标签

功能：

main-component 定义一个变量count,初始化为0，将count传递给son-component,son-component接收到数据显示在p标签中

main-component 在点击按钮时，实现对count的自增操作，要求son-component能够实时显示count对应的数据

son-component在接收到count之后，在count大于10的时候，将main-component的按钮禁用掉

（参考：<button v-bind:disabled="!isValid">clickMe</button>）

**Day4**

每日一练：

创建2个组件，main-component,son-component

视图：

main-component 显示一个按钮

son-component 显示一个p标签

功能：

main-component 定义一个变量count,初始化为0，将count传递给son-component,son-component接收到数据显示在p标签中

main-component 在点击按钮时，实现对count的自增操作，要求son-component能够实时显示count对应的数据

son-component在接收到count之后，在count大于10的时候，将main-component的按钮禁用掉

（参考：

<button v-bind:disabled="!isValid">clickMe</button>）

今天的安排：

1、SPA（single page application）概述

2、VueRouter来实现一个SPA的基本步骤

3、VueRouter来实现一个SPA的实现页面间的跳转

4、VueRouter来实现一个SPA的实现页面间的跳转的同时，完成参数的传递

一、SPA的概述

SPA（single page application）单页面应用程序，在一个完成的应用或者站点中，只有一个完整的html页面，这个页面有一个容器，可以把需要加载的代码片段插入到该容器中。

SPA的工作原理：

http://127.0.0.1/index.html#/start

①根据地址栏中url解析完整的页面：index.html

加载index.html

②根据地址栏中url解析#后的路由地址： start

根据路由地址，去在当前应用的配置中 找该路由地址的配置对象去查找该路由地址 所对应的模板的页面地址

发起异步请求加载该页面地址

③把请求来的数据加载到指定的容器中

二、通过VueRouter来实现一个SPA的基本步骤

①引入对应的vue-router.js

②指定一个盛放代码片段的容器

<router-view></router-view>

③创建业务所需要的各个组件

④配置路由词典

每一个路由地址的配置对象（要加载哪个页面...）

const myRoutes = [

{path:'/myLogin',component:TestLogin},

{path:'/myRegister',component:TestRegister}

]

const myRouter = new VueRouter({

routes:myRoutes

})

new Vue({

router:myRouter

})

⑤测试

在地址栏中 输入对应的不同的路由地址 确认是否能够加载对应的组件

练习：

需要大家创建一个SPA，这个SPA有3个组件，分别对应的是collect/detail/order

功能需求：

在地址栏中路由地址是:

/myColllect --> 收藏页组件

/myDetail --> 详情页组件

/myOrder --> 订单页组件

三、通过VueRouter来实现组件之间的跳转

提供了3种方式实现跳转：

①直接修改地址栏中的路由地址

②通过router-link实现跳转

<router-link to="/myRegister">注册</router-link>

③通过js的编程的方式

jumpToLogin: function () {

this.$router.push('/myLogin');

}

四、通过VueRouter来实现组件之间的跳转：参数的传递

login ---用户名---》main

①明确发送方和接收方

②配置接收方的路由地址

{path:'/myTest',component:TestComponent}

-->

{path:'/myTest/:id',component:TestComponent}

③接收方获取传递来的数据

this.$route.params.id

④跳转的时候，发送参数

this.$router.push('/myTest/20')

<router-link :to="'/myTest'+id">跳转</router-link>

练习：

视图：有两个组件

product-list ： 构造一个具有5条数据的数组，显示在列表中，每个列表项都有一个超链接

product-detail:p 显示字符串：这是id为\*\*的详情信息

功能：点击product-list的列表项，跳转到product-detail，并将数组中该元素的下标发送给product-detail，product-detail接收到数据显示在p标签。

①明确发送 接收

发送：list

接收：detail

②配置接收方的路由地址

{path:'/detail',} --》 {path:'/detail/:id',}

③接收方取值

this.$route.params.id

④发送参数

<router-link :to="'/detail/'+index"></router-link>

this.$router.push('/detail/20')

五、路由设置高级用法

alias 别名

{path:'/list',component:MyList,alias:'/lists'}

redirect 重定向

{path:'/productList',redirect:'/list'}

path:'\*' 异常处理

{path:'\*',component:'NotFound'}

六、路由嵌套

举个例子：

163邮箱为例，

login/mail(根据需求 加载inbox或者outbox)

login mail

--inbox

--outbox

路由嵌套的SPA实现的步骤：

A（/a）组件需要嵌套B组件(/b)和C组件(/c)

①准备嵌套其它组价的父组件 指定一个容器

在A组件指定一个容器

<router-view></router-ivew>

②在A组件的路由配置对象中指定children属性

{

path:'/a',

component:A,

children:[

{path:'/b',component:B},

{path:'/c',component:C},

]

}

补充：

//数字如果超出记录的次数，是不行的。

this.$router.go(num);

如果num是正数，向前进

如果num是负数，向后退

**Day5**

今日安排：

1、网络请求

2、通过vue-cli命令行方式所创建的模板项目 是如何进行编程的

3、综合小项目ToDoList

4、学子商场的后天管理界面框架搭建

fake api:

http://jsonplaceholder.typicode.com/posts

方便对于网络模块做测试。

正常情况，需要公司的服务器。

一、网络请求

需要vue-resource.js插件

this.$http

.get('url')

.then(function(response){

// response.data 就是服务器端所返回的数据

})

二、综合组件之间的通信

实现一个ToDoList。

①完成所有的组件的创建和使用

②add

点击add按钮时候，将用户输入的内容（todoinput），显示在（todolist）

核心代码：兄弟组件间通信

步骤1：var bus = new Vue()

步骤2：在准备接受数据的组件

bus.$on('addEvent',function(){

})

步骤3：触发事件

bus.$emit('addEvent',123)

将todolist中数组的元素 渲染在todoitem的每一个span标签。（父子组件通信）

③delete

在todoitem中点击delete按钮时，将该todoitem删除，由于todoitem的数量 取决于 todolist中数组

子组件 和 父组件通信、

三、基于Vue-Cli所创建的模板项目使用方式

好处：

①当代码修改，直接预览效果

hot reload

②框架已经搭建好了，快速进入到开发

③组织代码的方式

将组件通过.vue的文件 单独组织 放在src目录components

1、webstorm设置

为了提高webStrom在Vue项目中响应速度，设置：

选中项目，点击右键，mark directory as,excluded

进入到CLI：

windows+R

cmd

d:

cd vue\_cli

npm run dev

2、介绍单文件组件概念

文件扩展名为 .vue 的 single-file components(单文件组件) ；

有3个标记：template(是html) script（组件的设置） style（样式的设置）

3、vue项目启动流程

在执行npm run dev的时候，会去在当前文件夹下的项目中找package.json文件,启动开发服务器，默认端口是8080；

找到src的main.js文件，在该文件中new Vue的实例，要加载的模板内容App，App是src目录下的App.vue结尾的文件，在App.vue所对应的模板当中，有一个router-view

在src目录下有一个router文件夹，该文件夹有个index.js文件，该文件是配置路由词典，指定了路由地址是空，加载Hello组件

4、使用Vue的组件

步骤1：创建.vue结尾的文件

MyHeader.vue

步骤2：创建组件

在main.js文件中，

①引入要使用的组件

import MyHeader from '@/components/MyHeader'

②创建组件

Vue.component('my-header',MyHeader)

步骤3：使用组件

MyList.vue

<template>

<my-header> </my-header>

</template>

5、Vue项目 如何 部署在真正的服务器端

在工程所在的目录中，执行npm run build，

会生成一个dist的文件夹

就是我们可以直接部署（deploy）在服务器端的文件。

练习：

创建两个组件，my-login,my-main,要求，当路由地址是空或者'/login'的时候，访问my-login组件；

当路由地址是'/main'，访问my-main组件。

步骤1：

在components创建中创建MyLogin.vue,MyMain.vue

步骤2：

配置路由词典

参数传递：

①明确发送 接收

login main

②配置接收方路由地址

{path:'/main'}

-->

{path:'/main/:uName'}

③在接收方接收数据

this.$route.params.uName

④发送参数

this.$router.push('/main/michale')

重构todolist。

ToDoBox.vue

ToDoList.vue

TODoItem.vue

ToDoInput.vue

**Day6**

综合SPA练习：

有一个列表页 （my-list）

有一个input和按钮（添加：将用户的输入直接添加到当前的列表中）、一个列表（支持点击，当点击列表中的 列表项时，将当前序号 传递给详情页）

有一个详情页 (my-detail )

显示一个序号（从列表页传递过来的）

直接引入对应的js文件。

步骤:

①引.js vue-router.js)

②指定容器 <router入对应的js(vue-view></router-view>

③创建组件

④配置路由词典

⑤测试

⑥实现跳转功能

（①明确发送、接收 ②配置接收方 ③ 接收 ④发送）