正课:

1. 组件:
2. 组件生命周期:
3. 组件化开发
4. 组件间传参
5. 路由
6. 组件:

什么是: 拥有专属的HTML，CSS和数据的页面独立区域

为什么: 1. 便于重用

2. 松散耦合

3. 分工协作

何时: 今后，只要页面上一个独立的区域，都必须封装为组件，今后每个页面都是由多个组件组成的。

如何:

1. 创建Vue组件: 2步:
   1. 先在<template></template>中定义组件的HTML片段

要求<template>内只能有一个唯一的父元素包裹

* 1. 创建Vue组件

Vue.component(“组件名”,{

template:”选择器”, //找到页面上一个<template>元素，并监控其中的内容——组件的HTML片段

data(){ //每使用一次组件，就调用一次data函数，为当前组件生成一个新data对象

return {

模型变量

}

},

…同new Vue()

})

1. 使用组件: 一个组件，其实就是一个可重用的标签！组件名就是标签名

运行时: 用<template>中的HTML片段代替<组件名>标签的位置

2. 组件生命周期:

什么是: 一个组件的加载过程

包括: 4个阶段:

1. Create创建：创建组件实例对象，并创建data对象
2. Mount挂载：将组件模板中的元素，绑定后，挂载到DOM树
3. Update：模型变量被更新
4. Destory：要销毁一个组件

生命周期钩子函数:

什么是: 每个生命周期阶段抛出的事件处理函数

何时: 只要希望在某个生命周期阶段执行一项任务时，都要写在该阶段对应的构造函数中。

包括:

beforeCreate: 创建组件实例之前触发

created: 创建组件实例之后，但挂载组件到DOM树之前触发

new Vue($data:模型数据 $el:undefined)

已经可以发送ajax请求

暂时不能执行DOM操作！

beforeMount: 挂载组件到DOM树之前触发

mounted: 挂载组件到DOM树之后触发

new Vue($data:模型数据 $el:虚拟DOM树)

也可以发送ajax请求

可执行DOM操作！

beforeUpdate: 模型数据被更新前触发

updated: 模型数据更新后触发

beforeDestroy: 销毁组件对象之前触发

destroyed: 销毁组件后触发

3. 组件化开发:

组件: 页面中一个独立的，可重用的区域

如何进行组件化开发:

拿到页面后，先划分区域，每个区域都是拥有专有HTML，CSS，数据的独立组件

分为:

1. 根组件: new Vue()

通常一个页面，只有一个根组件

1. 全局组件: Vue.component()

什么是: 可在页面中任何位置使用的组件

1. 局部组件: 也称子组件:

什么是: 只能用于某个特定父组件内的组件

如何: 2步:

1. 将作为子组件的全局组件，降级为普通的对象

强调: 组件对象名应该是将来的组件标签名改为驼峰命名

1. 在父组件中添加components属性:{

子组件对象, …

}

强调: Vue会将驼峰命名的子组件，转化为-命名的标签名.

4. 组件间传值:

父子:

1. 父->子:

问题: 父组件的模型变量，子组件无权直接使用

解决: “属性下行”

如何: 2步:

1. 子: 定义属性:

子组件对象{

…

props:[“模型变量”,…]

}

说明: props中的模型变量:

对外: 可用作元素标签中的属性，让父组件可以通过绑定赋值。

对内: 等效于data{ … }

1. 父: 绑定属性，赋值:

<template id=”tpl父”>

<子组件 :模型变量=”父的模型变量”

2. 子->父: 事件上行

问题: 子组件无权直接删除父组件中的模型变量的内容

解决: 事件上行

如何: 2步:

1. 父: <template>  
    <子组件 @自定义事件=”父的处理函数”

父组件对象={

…

methods:{

处理函数(形参){ 实际修改父组件的模型变量}

}

}

1. 子: 子组件对象={

…

methods:{

事件处理函数(){

This.$emit(“自定义事件”,要给父的值)

}

}

}