正课:

1. 组件

2. 组件化开发

3. 组件的生命周期

4. SPA

1. 组件(Component):

什么是: 页面中一个块拥有专属的HTML，css，js和数据的独立功能区域。

为什么: 重用代码和功能，提高开发效率

何时: 只要页面中一块功能，可能被反复使用，就要定义为组件，然后反复使用组件。

如何：

1. 创建组件:

Vue.component("组件名",{

//每个小的组件内部，其实都是一个缩微的new Vue()。new Vue()中有什么！组件对象中就有什么！

~~el:"选择器",~~

template:`html片段，必须包含在一位唯一的父元素内`, //因为这段片段会被反复复制，最初的这一段片段，就称为模板。

~~data:{ ... },~~

data(){//函数: 每使用一次组件，就调用一次，就下一个新的蛋！

return {//相当于new Vue()中的data

变量

}

},

methods:{ ... },

watch:{ ... },

computed:{ ... },

...,

...,

})

2. 使用组件:

Vue中，组件其实就是一个可重用的自定义标签而已！组件名其实就是标签名！

运行时: 组件对象会用自己的template片段代替页面中组件标签所在的位置。

强调: 因为组件名会成为标签名！但是HTML不区分大小写！所有，组件名如果包含多个单词，不要用驼峰命名！应该用-分割多个单词，比如: todo-add todo-list todo-item my-counter my-header

2. 组件化开发:

什么是: 将一个大的页面先划分为多个小的组件，然后再分工协作开发。

为什么: 1. 便于多人分工协作！

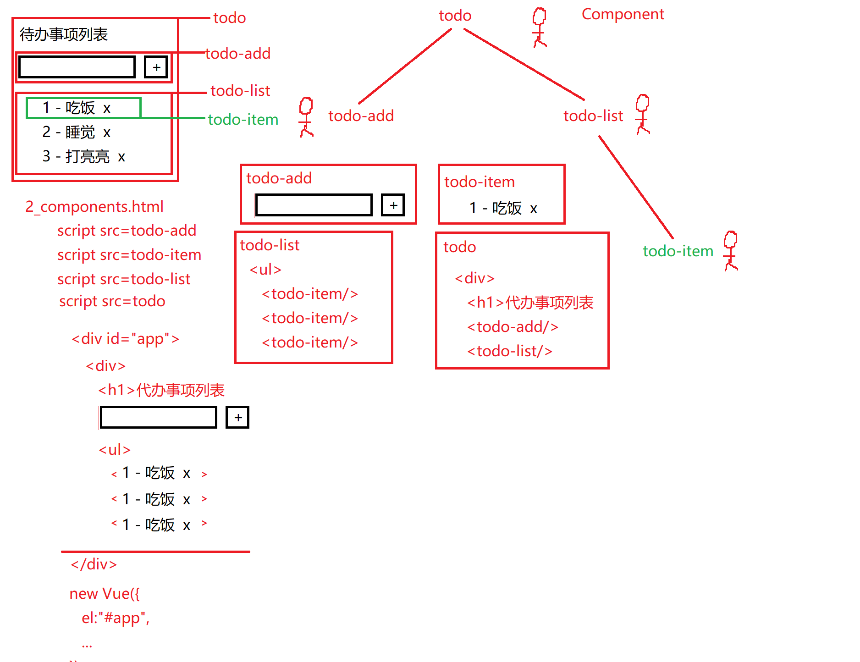
2. 松耦合

何时: 今后所有页面，都是划分成多个组件，分头开发的

如何:

1. 会划分组件: 按功能 或 区域

大组件内，还可进一步划分小的组件



2. 按照组件结构图，创建组件，引入组件，并在主界面中使用组件标签

1. 可以放在任何位置的组件——全局组件:

Vue.component(...)

问题: 可随意放在任何位置，不受限制。

2. 如果仅限于一个组件内使用的子组件——子组件

2步:

1. 将Vue.component()降级为一个普通的对象！

Vue.component("todo-item",{ ... })

降级 ↓

var todoItem={ //必须驼峰命名

template:`...`,

data(){ return { ... } },

methods:{ ... }

}

降级后，子组件对象名必须使用驼峰命名

2. 在父组件中，用components属性，包裹自己的子组件:

//全局组价可包含多个子组件

Vue.component("父组件",{

template:`...`,

data(){ return {...} },

...,

components:{ 子组件对象名, ... }

})

//子组件下还可以包含更子一级组件

var 父组件={

template:`...`,

data(){ return {...} },

...,

components:{ 子组件对象名, ... }

}

强调: 引入子组件时，也必须用驼峰命名放在components中引入。但是，放入components中的驼峰命名组件名，会被vue自动范围以-分割多个单词的形式。

比如: todoAdd -> <todo-add>

todoList -> <todo-list>

组件间传递数据: 3种:

1. 父->子:

问题: Vue中，父组件的成员，子组件无法直接使用！因为不是继承关系！而且组件的一切内容都是专属的独有的！

解决: 添加自定义属性: 2步:

1. 给要接收数据的子组件，添加自定义属性:

var 子组件={

template:`...`,

data(){

return {

...

}

},

props:[ "tasks" ]

}

结果: 将来凡是可以用子组件的地方：

<子组件 tasks="...">

而且: 自定义属性可用:绑定值！

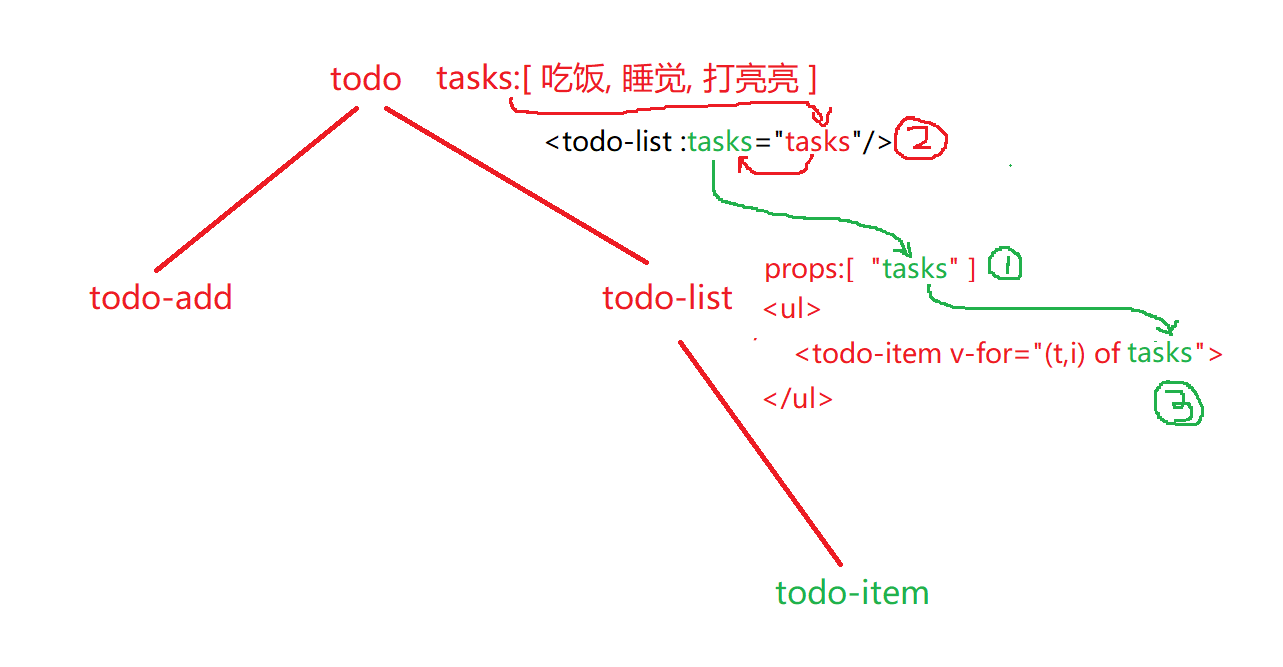
2. 父组件中，在子组件标签中，用:将父组件的变量值绑定给子组件的自定义属性。

<子组件 :tasks="tasks">

孩子的 爹的

结果: 在子组件中，props中自定义属性的用法和data中变量的用法完全一样！

比如: data中的变量可用于绑定语法或指令中。那么props中的自定义属性，也能用于绑定语法和指令中。



data vs props 区别：

data是当前组件专属的独有的，仅内部使用！

props可被父组件发现，并绑定数据。对外使用的！同时，对内又兼具data变量特点。

强调: 绑定属性值等效于赋值

如果绑定的是原始类型的值，则传下去的是副本。子组件中修改新变量，不影响父组件中的原变量。

如果绑定的是引用类型的对象，则传下去的是该对象的地址。子组件中修改新变量，等效于直接修改父组件中原对象的内容。

2. 子->父: 鄙视题视频中讲解

3. 子->子: 鄙视题视频中讲解

3. 组件的生命周期:

什么是: 一个组件加载过程中所经历的各个阶段

为什么: 在组件加载过程中，我们可能会自动执行一些操作。就好像“当页面加载完成后...”

何时: 只要希望在组件加载过程中，到某个阶段时，自动执行一件事。

包括: 4个阶段，包含8个钩子（回调）函数:

1. 创建create 必经阶段

创建组件对象

在组件对象内，创建data

有data了——可以对data执行操作了，比如发送ajax请求，初始化data中的变量

没有虚拟DOM树——不能执行DOM操作！

2. 挂载mount必经阶段

创建虚拟DOM树

将data中的数据，绑定到页面上

即有data，又有虚拟DOM树

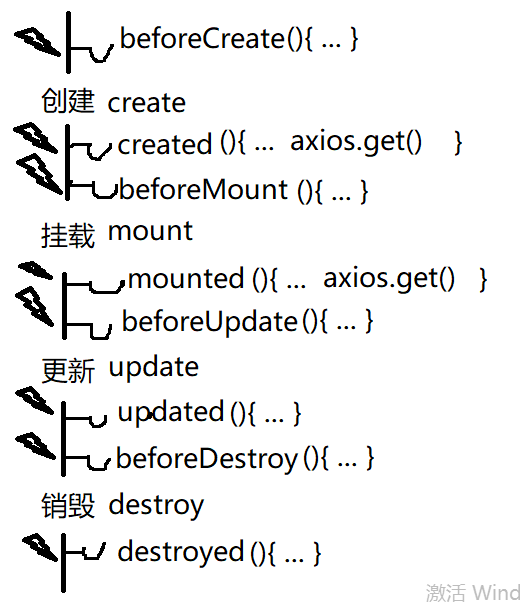
即可以发送ajax请求，初始化数据，又可以通过虚拟DOM树执行部分DOM操作。

=====================================================

3. 更新update 只在data中的变量更新后才进入update阶段

4. 销毁destroy 只在主动调用$destroy()函数后才进入销毁阶段。

八个钩子函数:



4. SPA: Single Page Application

单 页面 应用

什么是: 整个应用程序只有一个完整的.html页面。

所谓的页面其实是多个组件而已！

SPA应用有个本事:

监控地址栏

根据地址栏中相对路径的变化

替换不同的组件到html页面中指定区域

为什么:

|  |  |  |
| --- | --- | --- |
|  | 多页面应用 | 单页面应用 |
| 请求次数 | 每更换一次页面，都要重新请求服务器，增多了请求的次数。 | 开始时，将所有的组件下载到客户端本地。每次更换页面，只在客户端本地换组件。不需要服务器支持！极大的减少了请求的次数 |
| 共用的资源:  比如:  jquery.js bootstrap.css | 只要切换页面，就要反复请求共用的资源。 | 多个组件需要共用的资源，只在第一次请求页面是下载一次！之后切换组件，不用重新下载！ |
| 更新效率: | 多页面应用，每次更换页面要重建整棵DOM树——效率低！ | 每次切换组件，不需要删除原有整棵DOM树，只更新局部节点即可！ |
| 首屏加载速度 | 多页面应用，因为一次只请求一个页面，所以加载快 | 单页面应用需要在开始时，就下载所有组件到客户端备用。所以首屏加载慢  ——解决: 异步加载组件 |

何时: 今后几乎所有移动端的应用程序，都是单页面引用。

如何: 不切换页面，只切换页面的内容

借助Vue中的一个组件: vue-router

1. 创建整个项目唯一的html文件

在<div id="#app">中

用<router-view/>占位

2. 定义多个页面组件——都是子组件

3. 定义路由字典，创建路由器对象

什么是路由字典: 包含多个路径与组件对应关系的数组

什么是路由器: 可以自动监视地址栏变化，并根据路由字典的信息，自动选择对应的组件，替换页面区域的特殊的程序

4. 将router对象加入到new Vue()中

vue-router组件是采用锚点地址作为相对路径

路由跳转：

不要用a

用<router-link>代替:

<router-link to="/相对路径">文本</router-link>

自动翻译为: <a href="#/相对路径">文本</a>

路由传参:

1. 修改路由字典:

{

path:"/相对路径/:变量名",

component:组件名,

props:true //让变量自动成为组件中的props一员，就可直接绑定！

}

强调: 如果路由字典中一个路由带了参数，则跳转时，不带参数的地址，无法导航到这个组件！

2. 跳转时:

1. 写死的连接:

<router-link to="/相对路径/变量值">

强调: 变量值前！不要加:

2. 用程序跳转:

this.$router.push("/相对路径/变量值")

3. 在页面组件中，如何获得参数值:

1. props:[ "变量名" ]

变量名必须和路由字典中/:变量名一致！

共有的页头的处理: 全局组件: