第一章 Vue核心知识讲解

第一节 初识VUE，走进VUE的世界

简介：对比前端三大框架，讲述vue为什么这么火以及发展趋势

- 历史介绍
  - angular 09年，年份较早，一开始大家是拒绝 star:59.3k
  - react 2013年, 用户体验好，直接拉到一堆粉丝 star:119k
  - vue 2014年, 用户体验好 star:124k
- Vue等框架与jQuery的区别
  - jQuery是基于操作dom的库
  - Vue框架是以数据驱动和组件化开发为核心

第二节 引包、留坑、实例化 、插值表达式{{}}

简介：如何开展vue，用vue做出自己的第一个网页

1. 引包
   - 确认已经下载了node,然后执行命令 npm install vue (如需下载自己要的版本在vue后面加上@版本号)
   - 页面引入刚下载的包

     ```
     <script type="text/javascript" src="vue.js"></script>
     ```
2. 留坑 即留一个vue模板插入的地方或者是vue代码对其生效的地方
3. 实例化 即启动Vue

   启动： new Vue({el:目的地,template:模板内容});实例化传入的是一个对象options
   - options
     - 目的地 el 对应上面留坑的坑位，可通过id名，类名，标签名来查找 。方式和jq一样
     - 内容 template
     - 数据 data 值为函数形式也可是对象，但是都是用函数，因为用的函数最后也是return一个对象
4. 插值表达式{{ }}
   - 插值表达式内填入data里面的变量即可在页面取到变量值{{ data里的变量 }}

第三节 熟悉及使用常用指令

简介：展示vue的常用指令以及讲解使用场景

1. **什么是指令**
    - **在vue中提供一些对于页面+数据的更为方便的操作，这些操作就叫做指令。**
        - **譬如在HTML页面中这样使用**`<div v-xxx=''></div>`
    - **在vue中v-xxx就是vue的指令**
    - **指令就是以数据去驱动DOM行为的,简化DOM操作**
2. **常用的指令有哪些，及怎么使用这些指令**
    - **v-text 不可解析html标签**
    - **v-html 可解析html标签**
    - **v-if 做元素的插入（append）和移除（remove）操作**
    - **v-else-if**
    - **v-else**
    - **v-show display:none 和display：block的切换**
    - **v-for**
        - **数组 item，index**
        - **对象 value，key ，index**

第四节 阐述vue单双向数据流及事件绑定

简介：vue特色之数据的双向绑定和事件绑定详解

1. **vue单向数据流绑定属性值 v-bind: (属性) 简写 :(属性)**
    - **例子：**`<input v-bind:value="name" v-bind:class="name">`
        - **单向数据绑定 内存改变影响页面改变**
        - **v-bind就是对属性的简单赋值,当内存中值改变，还是会触发重新渲染**
2. **vue双向数据流 v-model 只作用于有value属性的元素**
    - **例子：**`<input v-model="name" v-bind:class="name">`
        - **双向数据绑定 页面对于input的value改变，能影响内存中name变量**
        - **内存js改变name的值，会影响页面重新渲染最新值**
3. **事件绑定v-on:事件名="表达式||函数名" 简写 @事件名="表达式||函数名"**
    - **事件名可以是原生也可以是自定义的**
4. **总结**
    - **v-model 双向数据绑定**
        - **vue页面改变影响内存（js）**
        - **内存（js）改变影响vue页面**

      v-bind 单向数据绑定只是内存（js）改变影响vue页面

第五节 过滤器

简介：讲述如何给数据添加一个管道进行进一步处理再输出

- **过滤器就是可以对我们的数据进行添油加醋然后再显示**
- **过滤器有全局过滤器和组件内的过滤器**
    - **全局过滤器Vue.filter('过滤器名',过滤方式fn);**
    - **组件内的过滤器 filters:{ '过滤器名',过滤方式fn }**
    - **{{ msg | 过滤器名}}**
- **最终都是在过滤方式fn里面return产出最终你需要的数据**
- **vue中的this是vue封装好给我们使用的，跟平常方法里面的this是不同的**

第六节 数据监听watch计算属性computed

简介：讲述数据的单个监听以及多个监听还有深度监听的不同

- **watch监听单个，computed监听多个**
    - **思考业务场景：**
        1. **类似淘宝，当我输入某个人名字时，我想触发某个效果**
        2. **利用vue做一个简单的计算器**
- **当watch监听的是复杂数据类型的时候需要做深度监听（写法如下）**
    - ```
      watch:{
                  msg:{
                    handler(val){
                     if(val.text=='love'){
                      alert(val.text)
                     }
                    },
                    deep:true//开启深度监听
                  }
                }
      ```

- **computed 监视对象,写在了函数内部, 凡是函数内部有this.相关属性,改变都会触发当前函数**

第二章 组件化开发知识介绍

第一节 组件化开发

简介：讲述页面拆分为组件进行开发和维护

- **创建组件的两种方式**
      ```

```
var Header = { template:'模板' , data是一个函数,methods:功能,components:子组件们 }//局部声明

Vue.component('组件名',组件对象);//全局注册 等于注册加声明了
```

- **组件类型**
  - **通用组件（例如表单、弹窗、布局类等)**
  - **业务组件（抽奖、机器分类）**
  - **页面组件（单页面开发程序的每个页面的都是一个组件、只完成功能、不复用）**
- **组件开发三步曲：声明、注册、使用**

第二节 slot插槽和ref、$parent

简介：讲述如何设计可扩展组件及获取子组件和父组件实例

- **slot插槽**
  - **slot就是子组件里给DOM留下的坑位**
  - **<子组件>DOM</子组件>**
  - **slot是动态的DOM**
- **ref获取子组件实例**
  - **识别：在子组件或元素上使用属性ref="xxxx"**
  - **获取：this.$refs.xxxx 获取元素**
  - **$el 是拿其DOM**
- $parent获取父组件实例（可在子组件直接使用this.$parent即可）

第三节 父子组件的通信

简介：搭起专属于父子组件之间的沟通桥梁

- **父传子**
  - **父用子的时候通过属性传递**
  - **子要声明props:['属性名'] 来接收**
  - **收到就是自己的了，随便你用**
    - **在template中 直接用**
    - **在js中 this.属性名 用**
- **子传父**
  - **子组件里通过$emit('自定义事件名',变量1，变量2)触发**
  - **父组件@自定义事件名='事件名'监听**
  - 子组件方法里 this.$emit('sendfather',val1,val2)触发自定义事件
    父组件里 <child @sendfather='mymethods'></child>

第四节 非父子组件之间的通信

简介：建立Bus总线机制实施非父子组件通讯

- **创建一个空实例（bus中央事件总线也可以叫中间组件）**
- **利用$emit $on的触发和监听事件实现非父子组件的通信**
    - ```
      Vue.prototype.$bus=new Vue()//在vue上面挂载一个$bus作为中央处理组件
      this.$bus.$emit('自定义事件名','传递的数据')//触发自定义事件传递数据
      this.$bus.$on('自定义事件名'，fn)//监听自定义事件获取数据
      ```
- **解决的方案还有vuex、provide/inject是解决同根往下派发、本地存储也可以进行非父子组件之间的通信**

第五节 vue的生命周期

简介：详述vue所有的生命周期钩子函数的作用

- **需要频繁的创建和销毁组件**
    - **比如页面中部分内容显示与隐藏，但是用的是v-if**
- **组件缓存**
    - **内置组件中**
    - **被其包裹的组件，在v-if=false的时候，不会销毁，而是停用**
    - **v-if="true" 不会创建,而是激活**
    - **避免频繁创建组件对象的性能损耗**
    - **组件的激活和停用**
        - **activated 和 deactivated**
- **成对比较**
    - **created 和 beforeCreate**
        - **A 可以操作数据 B 数据没有初始化**
    - **mounted 和 beforeMount**
        - **A 可以操作DOM B 还未生成DOM**
    - **updated 和 beforeUpdate**
        - **A 可以获取最终数据 B 可以二次修改**
      ```

- **destroyed 和 beforeDestroy**

  性能调优：频繁销毁创建的组件使用内置组件包裹

愿景："让编程不在难学，让技术与生活更加有趣" 更多教程请访问 xdclass.net

第三章 Vue核心插件之路由模块

第一节 路由的跳转原理（哈希模式）

简介：了解vue核心插件的工作原理

- **单页应用的路由模式有两种**
  - **哈希模式（利用hashchange 事件监听 url的hash 的改变）**
  - **history模式（使用此模式需要后台配合把接口都打到我们打包后的index.html上）**
- 哈希模式原理
  - ```
    window.addEventListener('hashchange', function(e) {
      console.log(e)
    })
    ```
- 核心是锚点值的改变，我们监听到锚点值改变了就去局部改变页面数据，不做跳转。跟传统开发模式url改变后 立刻发起请求，响应整个页面，渲染整个页面比路由的跳转用户体验更好

第二节 安装和使用路由

简介：引入路由插件，简单了解单页应用的实现

- **路由是以插件的形式引入到我们的vue项目中来的**
  - **vue-router是vue的核心插件**

    1:下载 `npm i vue-router -S`

    2:安装插件`Vue.use(VueRouter);`

    3:创建路由对象 `var router = new VueRouter();`

    4:配置路由规则 `router.addRoutes([路由对象]);`

    路由对象`{path:'锚点值',component:要(填坑)显示的组件}`

    5:将配置好的路由对象交给Vue

    - **在options中传递-> key叫做 router**
    ```

6:留坑(使用组件) `<router-view></router-view>`

第三节 路由的跳转

简介：简述路由的几种跳转方式及他们之间的异同

- **路由的跳转方式有：**
  - **通过标签：**`<router-link to='/login'></router-link>`
  - **通过js控制跳转**`this.$router.push({path:'/login'})`
- **区别：**

  this.$router.push() 跳转到指定的url，会向history插入新记录

  this.$router.replace() 同样是跳转到指定的url，但是这个方法不会向history里面添加新的记录，点击返回，会跳转到上上一个页面。上一个记录是不存在的。

  this.$router.go(-1) 常用来做返回，读history里面的记录后退一个

  vue-router中的对象：
  - **$route 路由信息对象,只读对象**
  - **$router 路由操作对象,只写对象**

第四节 路由的传参和取参

简介：详解路由之间的沟通交流

1. **查询参**
   - **配置（传参） :to="{name:'login',query:{id:loginid}}"**
   - **获取（取参） this.$route.query.id**
2. **路由参数**
   - **配置（传参） :to="{name:'register',params:{id:registerid} }"**
   - **配置路由的规则 { name:'detail',path:'/detail/:id'}**
   - **获取 this.$route.params.id**

- **总结：**

1. **:to传参的属性里 params是和name配对的 query和name或path都可以**
2. **使用路由参数必须要配置路由规则里面配置好参数名，否则刷新页面参数会丢失**

第五节 嵌套路由

简介：了解嵌套路由的使用场景和如何实现

1. **补充上一节知识点：js跳转路由传参和标签传参，路由相同而参数不同时页面不做刷新的问题**

   解决方案：`<router-view :key="$route.fullPath"></router-view>`

   - **代码思想**
     - **1:router-view的细分**
       - **router-view第一层中，包含一个router-view**
     - **2:每一个坑挖好了，要对应单独的组件**
     - **路由配置**
       -
```
routes: [
            {
                path:'/nav',
                name:'nav',
                component:Nav,
                //路由嵌套增加此属性
                children:[
                //在这里配置嵌套的子路由
                ]
            }
        ]
```
   - **案例**
     - **进入首页下面会有导航，个人中心、首页、资讯、我的之类的**

第六节 路由守卫

简介：了解路由守卫的作用和如何使用

```
const router = new VueRouter({ ... }
//前置的钩子函数 最后要执行next ( ) 才会跳转
router.beforeEach((to, from, next) => {
  // ...
})
//后置的钩子函数 已经跳转了不需要next
router.afterEach((to, from) => {
  // ...
})
```

- **主要是简单介绍一下，路由守卫主要用于检验是否登录了，没登录就跳转到登录页面不让他在其他页面停留，但是现在这种处理主要的都用请求的全局拦截来做了。大致了解一下路由守卫即可**

第四章 购物车实战

简介：购物车实战的代码实战与讲解

第四章为实战购物车只有源码没笔记，请参看源码学习！谢谢