03.组件化开发

学习要点：

1.开发思维

2.小实例演示

本节课我们来开始了解 Vue-cli 组件化开发的思路，然后演示小实例。

一. 开发思维

1. 在使用脚手架开发时，使用的是组件化的开发模式，也就是.vue 文件；
2. 一个.vue 文件就是一个组件，里面基本包含：html、js 和 css 三个部分；
3. 这种方式，分离了每个组件的关联，提高了内聚，所以叫：组件化开发；
4. 开发完毕后，再进行打包编译成常规模式即可运行；
5. 默认创建的项目，src 包含两个文件：App.vue 和 main.js；
6. 上节课了解过 App.vue 是根组件，而 main.js 是入口文件，注释如下；

```
//引入 Vue.js 框架
import Vue from 'vue'
//引入 App.vue 根组件
import App from './App.vue'

//去掉警告
Vue.config.productionTip = false

//根 Vue 实例(全局)
new Vue({
  /**
   *  Vue2.x 写法，配合了 ES6 箭头函数
   *  参考 API 手册得知，作用是渲染一个视图组件
   *  这里注册了根组件 App
   */
  render: h => h(App),
  /**
   *  $mount 方法，参考 API 手册得知是挂载函数
   *  相当于{el : '#app'}
   */
}).$mount('#app')
```

7. 下面，我们看下 App.vue 中，引入了一个测试组件：HelloWorld，我们先注释掉；
8. 取消掉测试组件后，查看页面元素或源代码，你会发现有一个基本的 html 结构；
9. 这个基本的 html 结构是脚手架默认提供的模板 public/index.html；
10. 而当我们引入这个测试组件后，在#app 里就添加了组件的全部内容；

二．小实例演示

1. 我们模仿脚手架给的测试组件，自行创建三个组件，并引入到根组件；

```
//App.vue
<template>
  <div id="app">
    <Header title="欢迎来到 Vue-cli"></Header>
    <Sidebar></Sidebar>
    <Footer></Footer>
  </div>
</template>

<script>
import Header from './components/Header.vue'
import Sidebar from './components/Sidebar'
import Footer from './components/Footer'

export default {
  name: 'App',
  components: {
    HelloWorld,
    Header,
    Sidebar,
    Footer
  }
}
</script>

//Footer.vue
<template>
    <p class="footer">版权所有，翻版必究</p>
</template>

<script>
    export default {
        name: "Footer",
    }
</script>

<style scoped>
    .footer {
        color: #2c3e50;
    }
</style>
```

```
//Sidebar.vue
<template>
    <ul class="list">
        <li v-for="item in link"><a href="#">{{item}}</a></li>
    </ul>
</template>

<script>
    export default {
        name: "Sidebar",
        data() {
            return {
                link : ['首页', '资讯', '图文', '关于']
            }
        }
    }
</script>

<style scoped>
    .list a {
        color: brown;
    }
</style>

//Header.vue
<template>
    <h2 class="title">{{title}}</h2>
</template>

<script>
    export default {
        name : "Header",
        props : {
            title : String
        }
    }
</script>
```

2. 本地服务器运行没问题，但打包部署后，出现空白，查阅错误代码是路径问题；
3. 由于是基于 Webpack，可以在根目录创建 vue.config.js，设置公共路径；

```
module.exports = {
    publicPath: './',
}
```