1.

v-bind动态绑定的本地图片src需要放在static这个文件夹中

2.

<!-- elementui组件的prop不是string类型的时候需要v-bind绑定 不然会报错 -->

3.

 <!-- props有标题、背景颜色等 通过一个对象传进去-->

    <index-content v-bind="content\_01"></index-content>

    <index-content v-bind="content\_02"></index-content>

 data() {

    return {

      content\_01: {

        title\_01: "第一阶段课程",

        title\_02: "Java设计和编程思想",

        container\_bg: "#fff",

        title\_bg: "url('/static/image/strong02.png')",

        title\_01\_col: "#000",

        title\_02\_col: "#000"

      }

}

4.

<!-- 通过props、v-bind、style和data里的样式对象 控制组件的样式(也可写行内style解决) -->

    <div class="index-content-container" :style="bgColor">

 data() {

        return {

            bgColor: {

                backgroundColor: this.container\_bg

            }

        }

    }

 props: {

        container\_bg: {

            type: String,

            default: '#fff'

        }

1. 使用/deep/或>>>深度选择器改变element-ui的默认样式；使用scoped 后，父组件的样式将不会渗透到子组件中，使用深度选择器则可以从父组件渗透到子组件

/deep/ .el-carousel\_\_indicator--horizontal:hover

              .el-carousel\_\_button

                transition: all 0.2s

                height: 7px

6.

### 二、关于子组件的根元素

　　使用了scoped属性之后，父组件的style样式将不会渗透到子组件中，然而**子组件的根节点元素**会**同时被设置了scoped的父css样式和设置了scoped的子css样式影响**，这么设计的**目的是父组件可以对子组件根元素进行布局**。

//子组件

<template>

    <div class="bottom-container">

        <p>Copyright </p>

    </div>

</template>

//调用子组件

<bottom></bottom>

//父组件中的样式修改子组件样式

.bottom-container

    background-color #ccc

    /deep/ p

            font-size: 30px

7.当使用了一些插件在ie会打不开，需要在webpack.base.conf.js 的include中做处理

{

        test: /\.js$/,

        loader: 'babel-loader',

        include: [resolve('src'), resolve('test'), resolve('node\_modules/webpack-dev-server/client'),

          //有些插件需要写进来兼容ie

          resolve('node\_modules/vue-baidu-map/components')

        ]

      }

1. es6语法转换成es5 安装babel-polyfill 并在main.js里

import 'babel-polyfill';

1. 在 vue 中，实现 Tab 切换主要有三种方式：使用动态组件，使用 vue-router 路由，使用第三方插件。

<https://blog.csdn.net/zhumengzj/article/details/79300784>

1. em和rem

原文链接：<http://caibaojian.com/rem-vs-em.html>

 rem 和 em 单位是由浏览器基于你的设计中的字体大小计算得到的像素值。

 em 单位基于使用他们的元素的字体大小。

 rem 单位基于 html 元素的字体大小。

 em 单位可能受任何继承的父元素字体大小影响

 rem 单位可以从浏览器字体设置中继承字体大小。

 使用 em 单位应根据组件的字体大小而不是根元素的字体大小。

 在不需要使用em单位，并且需要根据浏览器的字体大小设置缩放的情况下使用rem。

 使用rem单位，除非你确定你需要 em 单位，包括对字体大小。

 媒体查询中使用 rem 单位

 不要在多列布局中使用 em 或 rem -改用 %。

 不要使用 em 或 rem，如果缩放会不可避免地导致要打破布局元素。

10.vue的响应式

**对于已经创建的实例，Vue 不允许动态添加根级别的响应式 property**





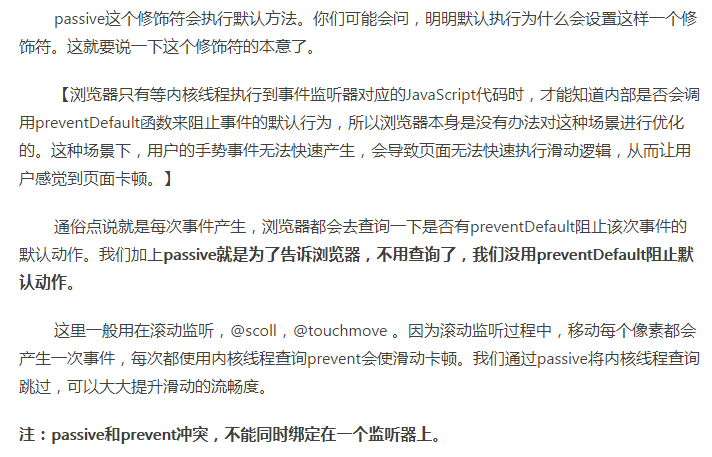
11.

虽然 Vue.js 通常鼓励开发人员使用“数据驱动”的方式思考，避免直接接触 DOM，但是有时我们必须要这么做。为了在数据变化之后等待 Vue 完成更新 DOM，可以在数据变化之后立即使用 Vue.nextTick(callback)。这样回调函数将在 DOM 更新完成后被调用。

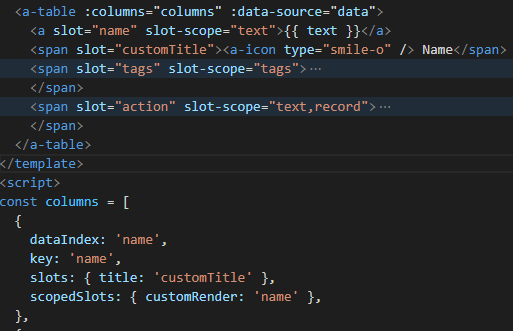


因为 $nextTick() 返回一个 Promise 对象

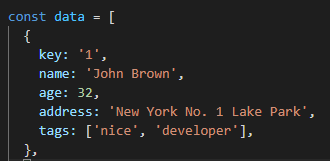
12. **.passive修饰符执行浏览器默认行为提高移动端性能**



13.ant design vue笔记之table



插槽slot=”name”对应columns中的scopedSolots，能渲染一整列，其中slot-scope中的第一个值”text”为一个变量，如果在data数组中的对象定义了name属性,text那就对应它的键值; slot-scope中的第二个值”record”为一个变量，代表data数组中的对象。



插槽slot=customTitle对应columns数组中对象的slots,能渲染tr中的th

14.在组件上用ref时用this.$refs获取到的是一个VueComponent对象，想通过操作dom改变样式使用this.$el.querySelector获取dom元素改变。

15.v-bind:style=”{}”绑定样式对象，对象中的属性会自动添加浏览器前缀

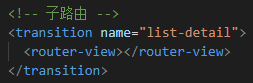




16.路由钩子之间执行的顺序

路由切换>全局beforeEach>组件beforeRouteEnter>全局afterEach>组件生命周期钩子>组件beforeRouteLeave

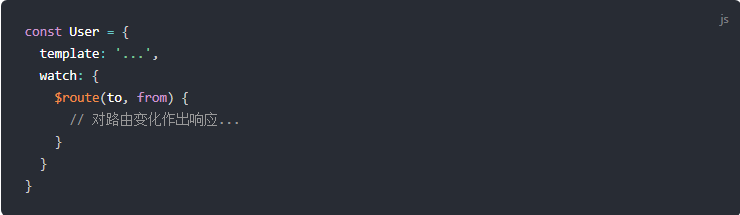
17.transition设置过渡效果，如果想设置某个组件过渡，transition需要包住router-view，如把组件设置成子路由，否则只能有离开过渡效果，没有进入过渡效果



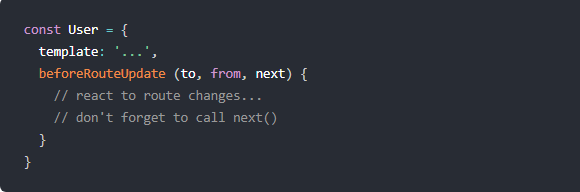
18.动态路由在同一组件切换时

提醒一下，当使用路由参数时，例如从 /user/foo 导航到 /user/bar，**原来的组件实例会被复用**。因为两个路由都渲染同个组件，比起销毁再创建，复用则显得更加高效。**不过，这也意味着组件的生命周期钩子不会再被调用**。

复用组件时，想对路由参数的变化作出响应的话，你可以简单地 watch (监测变化) $route 对象：



或者使用 2.2 中引入的 beforeRouteUpdate [导航守卫](https://router.vuejs.org/zh/guide/advanced/navigation-guards.html)：



19.props传回调函数

当传入回调作为props不能用es6的箭头函数,否则this为父组件实例,用es5则为子组件实例



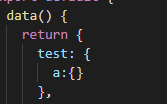
20.template中绑定data时报未定义错误

解决方案：

1）

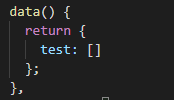
<!-- 如果test为空对象,会报未定义错误,因为test.a为undefined,不能获取undefined.b,需要把a初始化为空对象 -->

    <div>{{test.a.b}}</div>



2）v-for循环若list默认为空数组,当循环list，元素中为item.a.b,vue不会渲染这个列表，则不会报错。(已确认)





21. 父组件中插槽内容能够访问子组件中才有的数据，需要在子组件设置插槽prop，然后在父组件的template中使用v-slot:default=”slotProps”,使用为slotProps.test

父组件：

<template>

  <div>

    <slot-component>

      <template v-slot:aaa="slotProps">

        <div>

          <p>我是父组件的值:{{ test }}</p>

          <p>我是子组件的值:{{ slotProps.test }}</p>

        </div>

      </template>

    </slot-component>

  </div>

</template>

<script>

import SlotComponent from "./SlotComponent.vue";

export default {

  data() {

    return {

      test: "父组件data值",

    };

  },

  components: {

    SlotComponent,

  },

};

</script>

子组件：

<template>

    <div>

        <h1>插槽数据流</h1>

        <!-- <slot>默认内容</slot> -->

        <slot name='aaa' :test="childTest"></slot>

    </div>

</template>

<script>

export default {

    data() {

        return {

            childTest: '子组件数据'

        }

    }

}

</script>

22. slot-scoped

让父组件中插槽内容能够访问子组件中才有的数据，需要在子组件设置插槽prop，然后在父组件的template中使用slot-scoped(这个是旧语法新语法详见官网教程)

23. props的单向数据流

想修改引用型子组件prop,应在子组件另起一个data，赋了初始值为prop后(父组件的data改变,子组件的关联data并不会跟着改变)，修改子组件的prop时候要重新赋值一个新的引用型数据才不会影响父组件(或者用slice之后再赋值给子组件data最简单)。

若不想修改，只是根据prop做格式化输出,用computed。

24.子组件派发方法的$emit传参问题

## 前言

很多时候用$emit携带参数传出事件，并且又需要在父组件中使用自定义参数时，这时我们就无法接受到子组件传出的参数了。  
找到了两种方法可以同时添加自定义参数的方法。

## 方法一

子组件传出单个参数时：

// 子组件

this.$emit('test',this.param)

// 父组件

@test='test($event,userDefined)'

## 方法二

子组件传出多个参数时：

// 子组件

this.$emit('test',this.param1，this.param2, this.param3)

// 父组件 arguments 是以数组的形式传入

@test='test(arguments,userDefined)'

注：实参arguments对象可以用Array.from(arguments)或者扩展运算符[...arguments]或者Array.prototype.slice.call(arguments)方法转换成真正的数组。