Vue Router

官方文档：https://next.router.vuejs.org/zh/introduction.html

对vue-router的理解

vue 的一个插件库，专门用来实现 SPA，功能包括：

- 嵌套路由映射
- 动态路由选择
- 模块化、基于组件的路由配置
- 路由参数、查询、通配符
- 展示由 Vue.js 的过渡系统提供的过渡效果
- 细致的导航控制
- 自动激活 CSS 类的链接
- HTML5 history 模式或 hash 模式
- 可定制的滚动行为
- URL 的正确编码

对SPA的理解

1. 单页 Web 应用（single page web application，SPA）。
2. 整个应用只有一个完整的页面。
3. 点击页面中的导航链接不会刷新页面，只会做页面的局部更新。
4. 数据需要通过 ajax 请求获取

对路由的理解

什么是路由?

1. 一个路由就是一组映射关系（key - value）
2. key 为路径, value 可能是 function 或 component

路由分类

1. 后端路由：

   1) 理解：value 是 function, 用于处理客户端提交的请求。

   2) 工作过程：服务器接收到一个请求时, 根据请求路径找到匹配的函数 来处理请求, 返回响应数据。

2. 前端路由：

   1) 理解：value 是 component，用于展示页面内容。

   2) 工作过程：当浏览器的路径改变时, 对应的组件就会显示

路由使用快速入门

1.定义路由组件

views/Home.vue

```
<template>
  <div class="home">
    <h1>这里是主页</h1>
  </div>
</template>

<style scoped>
  .home{
    height: 100vh;
    background-color: antiquewhite;
  }
</style>
```

views/Login.vue

```
<template>
  <div class="login">
    <h1>这里是登录页</h1>
  </div>
</template>

<style scoped>
  .login{
    height: 100vh;
    background-color:aquamarine;
  }
</style>
```


2.定义路由

脚手架自动帮我们创建好了路由脚本 `src/router/index.js` 我们这里稍微修改下，定义自己的路由规则。

```
import { createRouter, createWebHashHistory } from 'vue-router'
import Home from '../views/Home.vue'
import Login from '../views/Login.vue' //导入路由组件

const routes = [      //定义路由 key-value
  {
    path: '/',        // key--路径
    name: 'Home',     // 路由的名称
    component: Home   // value--组件
  },
  {
    path: '/login',
    name: 'Login',
    component: Login
  }
]
// 创建路由对象并传递 `routes` 配置
const router = createRouter({
```

```
  // 内部提供了 history 模式的实现。为了简单起见，我们在这里使用 hash 模式。
  history: createWebHashHistory(),
  routes   // `routes: routes` 的缩写
})
//导出路由对象
export default router
```

3.使用路由

回到项目根目录下的`main.js`不用做任何更改，脚手架自动帮你调用了路由（前提是你没改默认的目录结构）

```
import { createApp } from 'vue'
import App from './App.vue'
import router from './router'  //导入路由
import store from './store'
// 自动调用了 use()
createApp(App).use(store).use(router).mount('#app')
```

回到根组件App.vue下，使用路由

```
<template>
  <!--使用 router-link 组件进行导航 -->
  <!--通过传递 `to` 来指定链接 -->
  <!--`<router-link>` 将呈现一个带有正确 `href` 属性的 `<a>` 标签-->
  <p>
    <router-link to="/">点我进入主页</router-link>
  </p>
  <p>
    <router-link to="/login">点我进入登录页</router-link>
  </p>
  <!-- 路由的出口 -->
  <!-- 路由匹配到组件会渲染在这里 -->
  <router-view></router-view>
</template>

<script>
export default {
  name: 'App',
}
</script>
```

router-link 会自动渲染成a标签，当点击时，自动在router-view将匹配的组件显示出来。

同样，在地址栏里之间输入地址也可以访问到页面。

通用组件与定制化组件

按照通用程度，我们可以把组件分为通用组件和定制化组件。我们在项目开发中使用一般很少从零开始开发组件，都是用别人写好的组件，我们对其稍微修改后做成符合产品特色的定制化组件，再封装到页面的过程。

Vue的特点就是组件化开发，意思就是一个一页面可以由多个组件构成，如主页有导航栏，测边栏，主要内容区，这些组件在其他页面也会被复用，如果重复开发会浪费大家时间也不利于维护，因此，开发页面就变成了开发组件再将这些组件拼到一起。

对于初学者来说，我们开发出的组件样式和兼容性都不够好，不符合生产需求，那么我们可以拿别人开发好的组件直接来用。优秀组件的集合就叫做组件库，目前和vue兼容性比较好的组件库有

饿了么团队推出的：组件 | Element (gitee.io)现以更新到element+,对应Vue3.x版本

有赞团队推出的：Vant - 轻量、可靠的移动端组件库 (gitee.io)现以支持Vue3.x版本

我们选择知名度更高的element-plus作为起点，等熟悉vue3操作之后，小伙伴们可以任意选择组件库，甚至自己开发组件库。

安装element-plus组件库

```
npm install element-plus --save
# 或者使用yarn
yarn add element-plus@^1.0.2-beta.54    # 截至文档编写时间 1.0.2-beta.55版本有BUG
```

这个只会安装到当前项目的node-models中，新建项目时要重新安装

引入element-plus组件库

首先检查vue-cli版本

```
## 查看@vue/cli版本，确保@vue/cli版本在4.5.0以上
vue --version
## 安装或者升级你的@vue/cli
npm install -g @vue/cli
```

完整引入

在 main.js 中写入以下内容：

```
import { createApp } from 'vue'
import App from './App.vue'
import router from './router'
import store from './store'

import ElementPlus from 'element-plus' //引入组件
import 'element-plus/lib/theme-chalk/index.css'; //引入样式

const app = createApp(App)
app.use(ElementPlus).use(store).use(router)
app.mount('#app')
```

如果想减小引入库体积，可以参考官网按需引入的方式。

案例开发-登录页面

接下来，我们使用element组件库完成一个登录页面

https://element-plus.gitee.io/#/zh-CN/component/form

登录页面最主要的组件就是表单和背景，我们可以对页面结构进行分解，实现各自组件，再拼积木一样拼起来。

登录表单

官网并没有一个现成的登录表单例子提供给我们，需要我们自己来组装。因此找到构成表单的元素（子组件）再将其评到一起就可以了。这其中包括了，输入框、登录按钮以及Form，每一个元素都有element的封装组件，拿过来用即可。

首先是输入框

分别为用户名输入框和密码输入框

饿了么输入框参考：https://element-plus.gitee.io/#/zh-CN/component/input

```
<template>
  <el-input v-model="account" placeholder='请输入用户名'></el-input>
  <el-input v-model="psw" placeholder='请输入密码' show-password></el-input>
  <h3>你输入的用户名:{{account}}</h3>
  <h3>你输入的密码:{{psw}}</h3>
</template>

<script>
import {ref} from 'vue'
export default {
  setup() {
    return {
      account: ref(''),
      psw: ref('')
    }
  },
}
</script>
```

记住，el-input 需要结合v-model双向绑定才能输入数据。

接下来是按钮

一个提交信息，一个清空输入框

饿了么按钮：https://element-plus.gitee.io/#/zh-CN/component/button

```
<template>
  <el-input v-model="account" placeholder='请输入用户名'></el-input>
  <el-input v-model="psw" placeholder='请输入密码' show-password></el-input>
  <h3>你输入的用户名:{{account}}</h3>
  <h3>你输入的密码:{{psw}}</h3>
  <el-button type="primary" @click="submit">提交</el-button>
  <el-button @click="reset">重置</el-button>
</template>

<script>
import {ref} from 'vue'

export default {
```

```
  setup() {
    let account= ref('')
    let psw= ref('')

    function submit(){
      account.value=''
      psw.value=''
      console.log('submit')
    }

    function reset(){
      account.value=''
      psw.value=''
      console.log('reset')
    }

    return {
      account,
      psw,
      submit,
      reset,
    }
  },
}
</script>
```

清除和提交暂时写个方法替代下，后续用element自己的。

再将其套入表单

表单参考：https://element-plus.gitee.io/#/zh-CN/component/form

参考自定义校验规则的样子来嵌套表单。

表单这里其实分为i两个元素：el-from 和 ele-form-item 前者嵌套后者

一个 ele-form-item里嵌套1个元素，两个按钮可以放一起

```
<template>
  <el-form>
    <el-form-item>
      <el-input v-model="account" placeholder='请输入用户名'></el-input>
    </el-form-item>
    <el-form-item>
      <el-input v-model="psw" placeholder='请输入密码' show-password></el-input>
    </el-form-item>
    <el-form-item>
      <el-button type="primary" @click="submit">提交</el-button>
      <el-button @click="reset">重置</el-button>
    </el-form-item>
  </el-form>
</template>
```

加上表单自身的属性

```
<template>
  <el-form :model="form" :rules="rules" ref="loginForm" label-width="60px">
```

```
    <el-form-item prop="account" label="账号">
      <el-input
        v-model="form.account"
        placeholder="请输入用户名"
        autocomplete="off"
      ></el-input>
    </el-form-item>
    <el-form-item prop="psw" label="密码">
      <el-input
        v-model="form.psw"
        placeholder="请输入密码"
        type="password"
        show-password
        autocomplete="off"
      ></el-input>
    </el-form-item>
    <el-form-item>
      <el-button type="primary" @click="submit">提交</el-button>
      <el-button @click="reset">重置</el-button>
    </el-form-item>
  </el-form>
</template>
```

加入校验规则，并返回相关数据

```
import { ref, reactive } from "vue";

export default {
  setup() {
    const form = reactive({
      account: "",
      psw: "",
    });
    const loginForm = ref(null);
    function submit() {
      form.account = "";
      form.psw = "";
      console.log("submit");
    }

    function reset() {
      loginForm.value.resetFields()
    }
    const rules = {
      account: [
        {
          required: true,
          message: "请输入用户名",
          trigger: "blur",
        },
      ],
      psw: [
        {
          required: true,
          message: "请输入密码",
          trigger: "blur",
        },
```

```
      ],
    };
    return {
      form,
      loginForm,
      submit,
      reset,
      rules,
    };
  },
};
```

给模板套上一些壳，再套入一些样式

```
<template>
  <div class="title">
    测试平台
  </div>
  <div class="login-form">
    <el-form :model="form" :rules="rules" ref="loginForm" label-width="60px">
    <div class="text-header">登录</div>
    <el-form-item prop="account" label="账号">
      <el-input
        v-model="form.account"
        placeholder="请输入用户名"
        autocomplete="off"
      ></el-input>
    </el-form-item>
    <el-form-item prop="psw" label="密码">
      <el-input
        v-model="form.psw"
        placeholder="请输入密码"
        type="password"
        show-password
        autocomplete="off"
      ></el-input>
    </el-form-item>
    <el-form-item>
      <el-button type="primary" @click="submit">提交</el-button>
      <el-button @click="reset">重置</el-button>
    </el-form-item>
  </el-form>
  </div>

</template>

<script>
import { ref, reactive } from "vue";

export default {
  setup() {
    const form = reactive({
      account: "",
      psw: "",
    });
    const loginForm = ref(null);
    function submit() {
```

```
      form.account = "";
      form.psw = "";
      console.log("submit");
    }

    function reset() {
      loginForm.value.resetFields()
    }
    const rules = {
      account: [
        {
          required: true,
          message: "请输入用户名",
          trigger: "blur",
        },
      ],
      pass: [
        {
          required: true,
          message: "请输入密码",
          trigger: "blur",
        },
      ],
    };
    return {
      form,
      loginForm,
      submit,
      reset,
      rules,
    };
  },
};
</script>

<style scoped>
.text-header {
  text-align: center;
  font-size: 20px;
  color: rgb(16, 16, 16);
  margin-bottom: 50px;
}
.login-form {
  position: absolute;
  width: 400px;
  height: 400px;
  top: 200px;
  right: 300px;
  border-radius: 10px;
  box-shadow: 1px 1px 5px #333;
  display: flex;
  justify-content: center;
  align-items: center;
}
.title {
  position: absolute;
  width: 400px;
  height: 80px;
```

```
  top: 50px;
  right: 300px;
  display: flex;
  justify-content: center;
  align-items: center;
  font-size: 60px;
  font-family: "Microsoft Yahei";
  color: rgb(13, 104, 139);
}
.el-form-item {
  color: black;
}
</style>
```

背景粒子效果

现在页面没有背景效果，我们可以搞个高大上一点的动态粒子背景效果。

首先背景可以作为一个单独的组件，后面再融合进登录页即可。

粒子效果这里我们可以采用`canvas-nest.js`

安装

```
# 使用 npm
npm install --save canvas-nest.js

# 或者使用 yarn
yarn add canvas-nest.js
```

使用

```
import CanvasNest from 'canvas-nest.js';
//color: 线条颜色, 默认: '0,0,0' ; 三个数字分别为(R,G,B)，注意用,分割
//pointColor: 交点颜色, 默认: '0,0,0' ; 三个数字分别为(R,G,B)，注意用,分割
//opacity: 线条透明度（0~1）, 默认: 0.5
//count: 线条的总数量, 默认: 150
//zIndex: 背景的z-index属性，css属性用于控制所在层的位置, 默认: -1
const config = {
    color: "18,156,255",
    opacity: 0.7,
    zIndex: -1,
    count: 150,
};

//获取element，模板元素需要加入ref="bg"属性
const bg = ref(null);
// 在 element 地方使用 config 渲染效果
const cn = new CanvasNest(element, config);

// destroy
cn.destroy();
```

完整的组件代码：background.vue

```
<template>
```

```
  <div ref="bg" class="bg"></div>
</template>

<script>
import { defineComponent, onBeforeUnmount, onMounted, ref,getCurrentInstance  }
from "vue";
import CanvasNest from "canvas-nest.js";

export default defineComponent({
  name: "background",
  setup() {
    const config = {
      color: "18,156,255",
      opacity: 0.7,
      zIndex: -1,
      count: 150,
    };
    const bg = ref(null);
    onMounted(() => {
      getCurrentInstance().cn = new CanvasNest(bg.value, config);
    });
    onBeforeUnmount(()=>{
      getCurrentInstance().cn.destroy() //销毁实例
    })
    return {
      bg,
    };
  },
});
</script>

<style scoped>
.bg {
  width: 100vw;
  height: 100vh;
}
</style>
```


最终效果

引用背景和登录表单 Login.vue

```
<template>
  <background />
  <div class="title">
    测试平台
  </div>
  <div class="login-form">
    <el-form :model="form" :rules="rules" ref="loginForm" label-width="60px">
    <div class="text-header">登录</div>
    <el-form-item prop="account" label="账号">
      <el-input
        v-model="form.account"
        placeholder="请输入用户名"
        autocomplete="off"
      ></el-input>
```

```
    </el-form-item>
    <el-form-item prop="psw" label="密码">
      <el-input
        v-model="form.psw"
        placeholder="请输入密码"
        type="password"
        show-password
        autocomplete="off"
      ></el-input>
    </el-form-item>
    <el-form-item>
      <el-button type="primary" @click="submit">提交</el-button>
      <el-button @click="reset">重置</el-button>
    </el-form-item>
  </el-form>
  </div>

</template>

<script>
import { ref, reactive } from "vue";
import background from '../components/Background'
export default {
  components:{
    background
  },
  setup() {
    const form = reactive({
      account: "",
      psw: "",
    });
    const loginForm = ref(null);
    function submit() {
      form.account = "";
      form.psw = "";
      console.log("submit");
    }

    function reset() {
      loginForm.value.resetFields()
    }
    const rules = {
      account: [
        {
          required: true,
          message: "请输入用户名",
          trigger: "blur",
        },
      ],
      pass: [
        {
          required: true,
          message: "请输入密码",
          trigger: "blur",
        },
      ],
    };
    return {
```

```
      form,
      loginForm,
      submit,
      reset,
      rules,
    };
  },
};
</script>

<style scoped>
.text-header {
  text-align: center;
  font-size: 20px;
  color: rgb(16, 16, 16);
  margin-bottom: 50px;
}
.login-form {
  position: absolute;
  width: 400px;
  height: 400px;
  top: 200px;
  right: 300px;
  border-radius: 10px;
  box-shadow: 1px 1px 5px #333;
  display: flex;
  justify-content: center;
  align-items: center;
}
.title {
  position: absolute;
  width: 400px;
  height: 80px;
  top: 50px;
  right: 300px;
  display: flex;
  justify-content: center;
  align-items: center;
  font-size: 60px;
  font-family: "Microsoft Yahei";
  color: rgb(13, 104, 139);
}
.el-form-item {
  color: black;
}
</style>
```