

Vue Router

Vue.js

Vue Router 是 Vue.js 的官方插件，用来快速实现单页应用。

Vue Router

- 单页应用
- 前端路由
- Vue Router

单页应用

Vue Router

SPA (Single Page Application) 单页面应用程序，简称单页应用。

指的是网站的 “所有” 功能都在单个页面
中进行呈现。

具有代表性的有后台管理系统、移动端、小程序等。

单页应用

- 优点：
 - 前后端分离开发，提高了开发效率。
 - 业务场景切换时，局部更新结构。
 - 用户体验好，更加接近本地应用。

单页应用

- 缺点：
 - 不利于 SEO。
 - 初次首屏加载速度较慢。
 - 页面复杂度比较高。

前端路由

Vue Router

前端路由，指的是 URL 与内容间的映射关系

URL、内容、映射关系。

前端路由

- Hash 方式
- History 方式

Hash 方式

前端路由

Hash 方式

- 通过 hashchange 事件监听 hash 变化，并进行网页内容更新。

```
<body>
  <div>
    <a href="#/">首页</a>
    <a href="#/category">分类页</a>
    <a href="#/user">用户页</a>
  </div>
  <div id="container">
    这是首页功能
  </div>
</body>
```

Hash 方式

- 通过 hashchange 事件监听 hash 变化，并进行网页内容更新。

```
window.onhashchange = function () {  
    var hash = location.hash.replace('#', '');  
    var str = ''  
    switch (hash) {  
        case '/':  
            str = '这是首页功能';  
            break;  
        case '/category':  
            str = '这是分类功能';  
            break;  
        case '/user':  
            str = '这是用户功能';  
            break;  
    }  
    document.getElementById('container').innerHTML = str;  
};
```


Hash 方式

前端路由

Hash 方式

- 封装以备复用。

```
var router = {  
  routes: {},  
  route: function (path, callback) {  
    this.routes[path] = callback;  
  },  
  init: function () {  
    var that = this;  
    window.onhashchange = function () {  
      var hash = location.hash.replace('#', '');  
      that.routes[hash] && that.routes[hash]();  
    };  
  }  
};
```

Hash 方式

- 封装以备复用。



```
var container = document.getElementById('container');  
// 定义路由  
router.route('/', function () {  
    container.innerHTML = '这是首页功能';  
});  
router.route('/category', function () {  
    container.innerHTML = '这是分类功能';  
});  
router.route('/user', function () {  
    container.innerHTML = '这是用户功能';  
});  
// 初始化路由  
router.init();
```

Hash 方式

- 特点总结：
 - Hash 方式兼容性好。
 - 地址中具有 #，不太美观。
 - 前进后退功能较为繁琐。

History 方式

前端路由

History 方式

- History 方式采用 HTML5 提供的新功能实现前端路由。

```
<body>
  <div>
    <a href="/">首页</a>
    <a href="/category">分类页</a>
    <a href="/user">用户页</a>
  </div>
  <div id="container">
    这是首页功能
  </div>
</body>
```

History 方式

- 在操作时需要通过 `history.pushState()` 变更 URL 并执行对应操作。

```
var router = {  
  routes: {},  
  route: function (path, callback) {  
    this.routes[path] = callback;  
  },  
  go: function (path) {  
    history.pushState(null, null, path);  
    this.routes[path] && this.routes[path]();  
  }  
}
```

History 方式

- 在操作时需要通过 `history.pushState()` 变更 URL 并执行对应操作。

```
var links = document.querySelectorAll('a');
var container = document.querySelector('#container');
links.forEach(function (ele) {
  ele.onclick = function (e) {
    var path = e.target.getAttribute('href');
    // 调用路由
    router.go(path);
    return false;
  }
});
```


History 方式

- 在操作时需要通过 `history.pushState()` 变更 URL 并执行对应操作。

```
// 定义路由规则
router.route('/', function () {
  container.innerHTML = '这是首页功能';
});
router.route('/category', function () {
  container.innerHTML = '这是分类功能';
});
router.route('/user', function () {
  container.innerHTML = '这是用户功能';
});
```

History 方式

前端路由

History 方式

- 前进后退功能，首先需要在更改 url 时保存路由标记。



```
go: function (path) {  
  history.pushState({ path: path }, null, path);  
  ...  
}
```

History 方式

- 通过 popstate 事件监听前进后退按钮操作，并检测 state。

```
init: function () {  
  var that = this;  
  window.addEventListener('popstate', function (e) {  
    var path = e.state ? e.state.path: '/';  
    that.routes[path] && that.routes[path]();  
  });  
}
```

History 方式

- 调用初始化方法监听前进后退操作并处理。



```
...  
router.init();  
...
```

Vue Router

Vue.js

是 Vue.js 官方的路由管理器，让构建单页面应用变得易如反掌。

Vue Router

- 基本使用
- 动态路由
- 嵌套路由
- 程式化导航

基本使用

Vue Router

基本使用

- 直接下载 / CDN
 - 最新版本: <https://unpkg.com/vue-router/dist/vue-router.js>
 - 指定版本: <https://unpkg.com/vue-router@3.4.9/dist/vue-router.js>

基本使用

- npm
 - `npm install vue-router`

基本使用

- Vue Router 提供了用于进行路由设置的组件 `<router-link>` 与 `<router-view>`。



```
<div id="app">
  <router-link to="/">首页</router-link>
  <router-link to="/category">分类</router-link>
  <router-link to="/user">用户</router-link>
  <router-view></router-view>
</div>
```

基本使用

- 定义路由中需要的组件，并进行路由规则设置。

```
var Index = {  
  template: '<div>这是首页的功能</div>'  
};  
var Category = {  
  template: '<div>这是分类的功能</div>'  
};  
var User = {  
  template: '<div>这是用户的功能</div>'  
};
```

基本使用

- 定义路由中需要的组件，并进行路由规则设置。



```
var routes = [  
  { path: '/', component: Index },  
  { path: '/category', component: Category },  
  { path: '/user', component: User }  
];
```

基本使用


- 创建 Vue Router 实例，通过 routes 属性配置路由。



```
var router = new VueRouter({  
  routes: routes  
});
```

基本使用

- 创建 Vue 实例，通过 router 属性注入路由。



```
var vm = new Vue({  
  el: '#app',  
  router: router  
});
```


命名视图

基本使用

基本使用

- 如果导航后，希望同时在同级展示多个视图（组件），这时就需要进行命名视图。



```
<div id="app">
  <router-link to="/">首页</router-link>
  <router-link to="/user">用户</router-link>

  <router-view name="sidebar"></router-view>
  <router-view></router-view>
</div>
```

基本使用

- 路由中通过 `components` 属性进行设置不同视图的对应组件。



```
var SideBar = {  
  template: `

这是侧边栏功能</div>`  
};


```



```
..  
{  
  path: '/',  
  components: {  
    sidebar: SideBar,  
    default: Index  
  }  
},  
..
```

动态路由

Vue Router

当我们需要将某一类 URL 都映射到同一个组件，就需要使用动态路由。

动态路由

- 定义路由规则时，将路径中的某个部分使用 `:` 进行标记，即可设置为动态路由。

```
var User = {  
  template: `<div>这是用户的功能</div>`  
};  
var routes = [  
  { path: '/user/:id', component: User }  
];
```

动态路由

- 设置为动态路由后，动态部分为任意内容均跳转到同一组件。

```
<div id="app">
  <router-link to="/user/1">用户1</router-link>
  <router-link to="/user/2">用户2</router-link>
  <router-link to="/user/3">用户3</router-link>
  <router-view></router-view>
</div>
```

动态路由

- : 部分对应的信息称为路径参数，存储在 `vm.$route.params` 中。

```
var User = {  
  template: `  
    <div>  
      这是用户 {{ $route.params.id }} 的功能  
    </div>`  
};
```


侦听路由参数

动态路由

侦听路由参数

- 如果要响应路由的参数变化，可以通过 watch 监听 \$route。

```
var User = {  
  template: `<div>这是 {{ $route.params.id }} 功能</div>`,  
  watch: {  
    $route (route) {  
      console.log(route);  
    }  
  }  
};
```

路由传参处理

动态路由

路由传参处理

- 这里通过路由的 props 设置数据，并通过组件 props 接收。

```
var routes = [  
  {  
    path: '/user/:id',  
    component: User  
  },  
  {  
    path: '/category/:id',  
    component: Category,  
    props: true  
  },  
];
```

路由传参处理

- 这里通过路由的 props 设置数据，并通过组件 props 接收。



```
var User = {  
  template: `<div>这是 {{ $route.params.id }} 的功能</div>`  
};  
var Category = {  
  props: ['id'],  
  template: `<div>这是 {{ id }} 功能</div>`  
};
```

路由传参处理

动态路由

路由传参处理

- 包含多个命名视图时，需要将路由的 props 设置为对象。



```
var SideBar = {  
  template: `<div>这是侧边栏功能</div>`  
};
```



```
{  
  path: '/category/:id',  
  components: {  
    default: Category,  
    sidebar: SideBar  
  },  
  props: {  
    default: true,  
    sidebar: false  
  }  
}
```

路由传参处理

- 如果希望设置静态数据，可将 `props` 中的某个组件对应的选项设置为对象，内部属性会绑定给组件的 `props`。

路由传参处理

```
var SideBar2 = {  
  props: ['a', 'b'],  
  template: `  
    <div>  
      这是右侧侧边栏功能: {{ a }} {{ b }}  
    </div>  
  `,  
};
```

```
{  
  path: '/category/:id',  
  components: {  
    default: Category,  
    sidebar: SideBar,  
    sidebar2: SideBar2  
  },  
  props: {  
    default: true,  
    sidebar: false,  
    sidebar2: { a: '状态1', b: '状态2' }  
  }  
},
```

嵌套路由

Vue Router

实际场景中，路由通常由多层嵌套的组件组合而成，这时需要使用嵌套路由配置。

嵌套路由

- 使用 `children` 来进行嵌套路由中的子路由设置。

嵌套路由

```
var routes = [  
  {  
    path: '/user',  
    component: User,  
    children: [  
      {  
        path: 'hobby',  
        component: UserHobby  
      },  
      {  
        path: 'info',  
        component: UserInfo,  
        children: [  
          { path: 'age', component: UserInfoAge },  
          { path: 'school', component: UserInfoSchool },  
        ]  
      }  
    ]  
  }  
];
```

编程式导航

Vue Router

编程式导航，指的是通过方法设置导航。

程式导航

- `router.push()` 用来导航到一个新 URL。



```
vm.$router.push('/user');  
vm.$router.push({path: '/user'});  
vm.$router.push({path: '/user/123'});
```


程式导航

- `<router-link>` 的 `to` 属性使用绑定方式时也可属性对象结构。



```
<router-link :to="{ path: '/user/10' }">用户10</router-link>
```

命名路由

编程式导航

命名路由

- 设置路由时添加 name 属性。

```
var School = {  
  template: '<div>School 组件: {{ $route.params }}</div>'  
};  
var routes = [  
  {  
    path: '/user/:id/info/school',  
    name: 'school',  
    component: School  
  }  
];
```

命名路由

- 在 `push()` 中通过 `name` 导航到对应路由，参数通过 `params` 设置。



```
vm.$router.push({ name: 'school', params: { id: 20, demo: '其他数据' }});
```

命名路由

- 也可以在 `<router-link>` 中使用。



```
<router-link :to="{ name: 'school', params: { id: 1 } }">用户学校</router-link>  
<router-link :to="{ name: 'school', params: { id: 2 } }">用户学校</router-link>  
<router-link :to="{ name: 'school', params: { id: 3 } }">用户学校</router-link>
```

重定向

其他功能

重定向

- 示例如下：



```
var routes = [  
  { path: '/', component: Index },  
  { path: '/category/:id', component: Category },  
  { path: '/category', redirect: '/' }  
];
```

别名

其他功能

别名

- 示例如下：

```
var routes = [  
  {  
    path: '/user/:id/info/school/intro/:date',  
    name: 'school',  
    component: School,  
    alias: '/:id/:date'  
  }  
];
```

别名

- 示例如下：



```
<router-link :to="{ name: 'school', params: { id: 1, date: 0101 } }">用户学校</router-link>  
<router-link to="/10/0612">用户学校</router-link>
```

导航守卫

其他功能

导航守卫

- 示例如下：



```
router.beforeEach(function (to, from, next) {  
  console.log(to, from);  
  next();  
});
```

History 模式

其他功能

History 模式

- 需要通过 Vue Router 实例的 mode 选项来设置，这样 URL 会更加美观，但同样需要后端支持避免问题。



```
var router = new VueRouter({  
  mode: 'history',  
  routes: [  
    ...  
  ]  
});
```

拉勾教育

— 互联网人实战大学 —



下载「拉勾教育App」
获取更多内容