

# Vue Router

# Need A Router?

- Many applications do not need router features
- Signs a router is needed
  - want URL to change when different “pages” (or views) are rendered so they can be bookmarked and identified by search engines
  - need non-trivial routing logic including actions before or after route changes and conditionally enabling some route changes
  - need to provide data to components rendered by new routes
  - have a large number of pages
- If needed, often only basic features are needed



# Non-router Approach

- Store current route name in application state
  - such as in Vuex which is covered in next section
- Change route by modifying route name in state
- Test current route name using directives **v-if**, **v-else-if**, and **v-else** to select component to render
  - typically done in top component

```
<template>
  <div id="app">
    <About v-if="route === 'about'" />
    <Products v-else-if="route === 'products'" />
    <Home v-else />
  </div>
</template>

<script>
import {mapState} from 'vuex';
import store from './store';
import About from './components/About.vue';
import Home from './components/Home.vue';
import Products from './components/Products.vue';

export default {
  name: 'App',
  components: {About, Home, Products},
  computed: {
    ...mapState({
      route: state => state.route
    })
  },
  store
};
</script>
```

# Vue Router Overview

- Most popular routing library for Vue
- Developed by and maintained by Vue team
- Maps URLs to components
- Supports navigation between “pages” using `<router-link>` elements
- Current route can also be changed from JavaScript code
- Current route is rendered in a `<router-view>` element
  - typically rendered by top-most component



# Installing

- When creating a project with Vue CLI, select “Router” feature
  - will start with two configured routes named “home” and “about”
- When not using Vue CLI, install with **`npm install vue-router`**

# Setup ...

- Route configuration is typically done in `src/router.js`
- When using Vue CLI and "Router" feature is selected, this file will be created
- When not created through Vue CLI, create this file manually
- Content should be similar to the following

```
import Vue from 'vue';
import Router from 'vue-router';
// Import app-specific view components here.

Vue.use(Router);

export default new Router({
  mode: 'history', // doesn't use "hash routing"
  base: process.env.BASE_URL, // defaults to /
  routes: [ // array of route definition objects
    {path: '/some-url', component: SomeComponent},
    // more route definition objects go here
  ]
});
```

more on this later

usually not needed

path is part of URL  
after domain



# ... Setup

- `src/main.js` should import `Router` object exported from `router.js` and configure Vue to use it as follows

```
import Vue from 'vue';
import App from './App.vue'; // top component
import router from './router';

new Vue({
  router, // tells Vue to use router
  render: h => h(App)
}).$mount('#app'); // "app" is id specified in public/index.html
```

"h" is short for "hyperscript",  
but is really the  
`createElement` function

# Rendering Route Components

- Top component, typically defined in `src/App.js`, is common place to create links to routes
- If `<a>` tags are used, each click on them will result in a call to the server and loss of application state
- Using `<router-link>` elements instead avoids this and processes route changes entirely in browser
- Example

```
<template>
  <div id="app">
    <nav>
      <router-link to="/">Home</router-link>
      <router-link to="/page1">Page 1</router-link>
      <router-link to="/page2">Page 2</router-link>
    </nav>

    <router-view/>
  </div>
</template>
```

route components will render here



# Directory Structure

- Some Vue apps place components associated with routes in `src/view` directory instead of `src/components`
- Not required



# Route Data

- In route definition objects,  
**path** string values can contain colon-prefixed parts to allow data to be passed when route is changed
  - example: `/fruit/:name/:size`
- In components rendered by a route change
  - **route parameters** are stored in `this.$route.params`
    - an object where keys are names after colons and values are their values
    - example path is `/fruit/banana/small`
  - **query parameters** are stored in `this.$route.query`
    - an object where keys are query parameter names and values are their values
    - example path is `/fruit/banana?color=yellow&size=small`
  - **hash value** is stored in `this.$route.hash`
    - a string that is the value including a leading #
    - example path is `/fruit/banana#uncommon`



# Route to Same Component

- When navigating to a route that is same as current route, but with different route parameters
  - same component will be used
  - lifecycle methods for creating and mounting the component will not be called again
- A **watch** on **\$route** property can be used to do things after a route parameter change
- Example

```
watch: {  
  $route(to, from) {  
    // Do things after route parameter change.  
  }  
}
```

# Route Change From Code

- Component methods can change current route by calling `this.$router.push(newRouteUrl)`
  - pushes new URL on to history stack so browser back button can be used to return to previous URL
  - example ahead
- Alternatively call `this.$router.replace(newRouteUrl)`
  - replaces current URL at top of the history stack so browser back button will not return to previous URL



# Wildcards

- To match any URL not matched by another route, use `*` for path
- `*` can also be used as a wildcard in any path part
- Matching string is held in `$route.params.pathMatch`
- Regular expressions can also be used in route paths
- It is possible for a URL to match more than one route path
  - when this happens, the first matching route path in the order defined is selected



# Hash Routing

- Technique where URLs contain # character
- Changes in URL before hash are handled by server
- Changes after hash are handled in browser
- By setting router **mode** to **history** (as was done earlier) the same functionality is achieved without hash characters which results in better looking URLs
  - default mode is "**hash**" which uses hash routing
- Use of hash routing is not recommended



# Router Example ...

```
import Vue from 'vue';
import App from './App.vue';
import router from './router';

Vue.config.productionTip = false;

new Vue({
  router,
  render: h => h(App)
}).$mount('#app');
```

main.js

**prevents following message** in devtools console on startup:  
"You are running Vue in development mode.  
Make sure to turn on production mode when deploying for production.  
See more tips at <https://vuejs.org/guide/deployment.html>"

```
import Vue from 'vue';
import Router from 'vue-router';
import Fruit from './views/Fruit.vue';
import Home from './views/Home.vue';

Vue.use(Router);

export default new Router({
  mode: 'history',
  routes: [
    {path: '/fruit/:name', component: Fruit},
    {path: '*', component: Home}
  ]
});
```

router.js

```
<template>
  <div class="Home">
    This is the home page.
  </div>
</template>
```

Home.vue

# ... Router Example ...

```
<template>
  <div class="app">
    <nav>
      <router-link to="/">Home</router-link>
      <router-link to="/fruit/apple">Apple</router-link>
      <router-link to="/fruit/orange">Orange</router-link>
      <button @click="showFruit('grapes')">Grapes</button>
      <button @click="showFruit('kiwi')">Kiwi</button>
      <button @click="showFruit('strawberry')">Strawberry</button>
    </nav>
    <router-view />
  </div>
</template>

<script>
export default {
  methods: {
    showFruit(name) {
      // This demonstrates routing from code
      // instead of from a <router-link>.
      this.$router.push('/fruit/' + name);
    }
  }
};
</script>

<style scoped>
  ...
</style>
```

App.vue

[Home](#) [Apple](#) [Orange](#)

Fruit: apple





# ... Router Example ...

```
<template>
  <div>
    <div>Fruit: {{ name }}</div>
    
  </div>
</template>

<script>
export default {
  computed: {
    imageUrl() {
      return `/images/${this.name}.jpeg`;
    },
    name() {
      return this.$route.params.name;
    }
  },
  beforeRouteUpdate(to, from, next) {
    const name = to.path.split('/')[2];
    if (name === 'kiwi') {
      alert('No kiwi please!');
    } else {
      next(); // allows navigation
    }
  },
}
```

Fruit.vue

Vue Router adds  
"in-component route guards"  
including this, **beforeRouteEnter**, and  
**beforeRouteLeave**.

comes from :name part  
of route path on slide 15

# ... Router Example

```
watch: {
  // After route change ...
  $route(to, from) {
    console.log(
      'Fruit.vue watch $route: switched from',
      from.path,
      'to',
      to.path
    );
    const {name} = this.$route.params;
    console.log('Fruit.vue watch $route: name =', name);
  }
};
</script>

<style scoped>
img {
  height: 200px;
}
</style>
```

Fruit.vue



# Per Route Guards

- Alternative to “in-component route guards” in previous example
- Can add **beforeEnter** method to route definition objects
  - like those on slides 6 and 15
- Takes **to**, **from**, and **next** parameters
  - **to** and **from** are **Route** objects that describe current and target routes
  - **next** is a function that must be called to allow navigating to route
  - if **next** is not called, navigation will not take place

# Styling

- `<a>` element produced by `<router-link>` of active route is given CSS class name **`router-link-exact-active`**
- Can use to style current route link differently from others
- Example

```
<template>
  <div class="app">
    <nav>
      <router-link to="/">Home</router-link>
      <router-link to="/fruit/apple">Apple</router-link>
      <router-link to="/fruit/orange">Orange</router-link>
    </nav>
    <router-view />
  </div>
</template>

<script>
...
</script>

<style scoped>
nav a.router-link-exact-active {
  color: green;
}
</style>
```



# Accessing Routes



- All components are injected with the properties `$router` and `$route`
- `this.$router` is a `Router` object
  - described on next slide
- `this.$route` is a `Route` object
  - described in two slides



# Router Objects



- Properties

- **app** - root Vue instance
- **mode** - mode that router is using; supported values are:
  - **hash** - uses hash routing
  - **history** - uses HTML5 History API
  - **abstract** - useful for server-side rendering
- **currentRoute** - current route

- Methods - "global navigation guards"

- **beforeEach**((to, from, next) => { ... })
- **beforeResolve**((to, from, next) => { ... })
  - "called right before the navigation is confirmed, after all in-component guards and async route components are resolved"
- **afterEach**((to, from) => { ... })

to and from parameters are **Route** objects; **next** is a function that should be called to allow navigation to proceed



# Route Objects



- Properties

- **path** - absolute path URL
- **fullPath** - like **path**, but includes query parameters and hash, if any
- **name** - name of route if named
- **params** - an object where keys are path parameter names and values are their values
- **query** - an object where keys are query parameter names and values are their values
- **hash** - part of **fullPath** that starts with and includes #
- **redirectedFrom** - name of route redirected from, if any
- **matched** - array containing descriptions of nested routes; advanced feature

- Methods

- none



# Named and Nested Routes



- Vue Router also supports
  - named routes
  - named views
  - nested named views
- These are advanced topics that most applications do not need



# For More On vue-route

- See <https://router.vuejs.org/guide/>

# Exercise ...

- Add use of Vue Router in dog app ...
- `npm install vue-router`
- Create `src/router.js` →
- Create `src/components/About.vue` ↓

```
import Vue from 'vue';
import Router from 'vue-router';
import About from './components/About.vue';
import Dogs from './components/Dogs.vue';

Vue.use(Router);



export default new Router({
  mode: 'history', // doesn't use hash routing
  routes: [
    {path: '/about', component: About},
    {path: '/dogs', component: Dogs},
    {path: '*', redirect: '/about'}
  ]
});
```

router.js

About Dogs

This app was created in order to learn about the Vue Framework. It first appeared at MidwestJS 2019.

About Dogs

Name	Actions
Eddie	
Snoopy	

Name

```
<template>
  <div>
    This app was created in order to
    learn about the Vue Framework.
    It first appeared at MidwestJS 2019.
  </div>
</template>
```

About.vue



# ... Exercise ...

- Modify `main.js`
  - `import router from './router';`
  - add `router` property in object passed to `new Vue()`

```
new Vue({  
  router,  
  render: h => h(App)  
}).$mount('#app');
```

# ... Exercise

- Modify **App.vue**

- replace `<Dogs />` in template with this →
- can remove this → since components used are now specified in `router.js`
- add this styling → and ask questions if you don't understand it

```
<nav>
  <router-link to="/about">About</router-link>
  <router-link to="/dogs">Dogs</router-link>
</nav>
<router-view />
```

```
components: {
  Dogs
}
```

```
#app > div {
  padding: 20px;
}

body {
  margin: 0;
}

nav {
  border-bottom: solid gray 1px;
  padding: 20px;
}

nav a {
  font-size: 24px;
  margin-right: 20px;
  text-decoration: none;
}

nav a.router-link-exact-active {
  color: green;
}
```