Web Programming

TODAY'S TOPICS

- Vue Router
  - Dynamic Route Matching
  - Nested Routes
  - Programmatic Navigation
- Linking front-end with back-end
  - Let's update the "youblog" website

DYNAMIC ROUTE MATCHING

In vue-router we can use a dynamic segment in the path to achieve that:

Now URLs
like /user/foo and /user/bar
will both map to the same route.

```
const User = {
  template: '<div>User</div>'
}

const router = new VueRouter({
  routes: [
    // dynamic segments start with a colon
    { path: '/user/:id', component: User }
  ]
})
```

DYNAMIC ROUTE MATCHING

You can have multiple dynamic segments in the same route, and they will map to corresponding fields on $route.params. Examples:

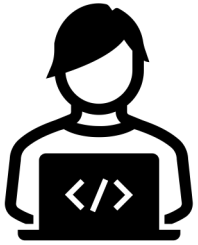
pattern	matched path	$route.params
/user/:username	/user/evan	`{ username: 'evan' }`
/user/:username/post/:post_id	/user/evan/post/123	`{ username: 'evan', post_id: '123' }`

$ROUTE.PARAMS

When a route is matched, the value of the dynamic segments will be exposed as this.$route.params in every component.

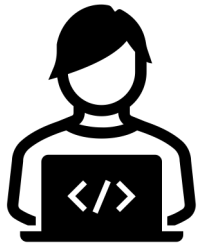
```
const User = {
  template: '<div>User {{ $route.params.id }}</div>'
}
```

$route.query:
An object that contains key/value pairs of the query string.
For example, for a path /foo?user=1, we get $route.query.user = 1. If there is no query the value will be an empty object.



Code Example
Click Here!

NESTED ROUTES



Code Example
Click Here!

```
const router = new VueRouter({
  routes: [
    {
      path: '/user/:id',
      component: User,
      children: [
        {
          // UserProfile will be rendered inside User's <router-view>
          // when /user/:id/profile is matched
          path: 'profile',
          component: UserProfile
        },
        {
          // UserPosts will be rendered inside User's <router-view>
          // when /user/:id/posts is matched
          path: 'posts',
          component: UserPosts
        }
      ]
    }
  ]
})
```

Root path

Root component

Children

PROGRAMMATIC NAVIGATION

Aside from using <router-link> to create anchor tags for declarative navigation, we can do this programmatically using the router's instance methods.

```
this.$router.push(location, onComplete?, onAbort?)
```

To navigate to a different URL, use this.$router.push. This method pushes a new entry into the history stack, so when the user clicks the browser back button they will be taken to the previous URL.

```
// literal string path
router.push('home')

// object
router.push({ path: 'home' })

// named route
router.push({ name: 'user', params: { userId: '123' } })

// with query, resulting in /register?plan=private
router.push({ path: 'register', query: { plan: 'private' } })
```

NAMED ROUTES

Sometimes it is more convenient to identify a route with a name, especially when linking to a route or performing navigations. You can give a route a name in the routes options while creating the Router instance:

```
const router = new VueRouter({
  routes: [
    {
      path: '/user/:userId',
      name: 'user',
      component: User
    }
  ]
})
```

```
<router-link :to="{ name: 'user', params: { userId: 123 }}">User</router-link>
```

LET'S UPDATE THE "YOUBLOG" WEBSITE

TODAY'S EXERCISE

Tutorial

- Home page (search)
- Blog detail page
- Create new blog
- Delete blog
- Add new comment

Exercise

- Like comment
- Delete comment
- Update comment
- Update blog