Updating Page Title & Metadata with Vue.js & vue-router

alligator.io/vuejs/vue-router-modify-head

vue-router is an excellent routing solution for Vue.js, but there's one thing that it doesn't handle that can drive developers (especially those experienced with SEO) mad. That is the process of updating the page title and metadata on route change. Now it makes sense that vue-router doesn't mess with this. Vue tries to stay out of the `head` and `body` elements as much as possible, as they can be rather volatile. But often times you'll want the title of the browser to change when the page changes, right? And you wouldn't want every search result or link to your website to say "Home Page" for all routes. So let's take a look at how to add this feature ourselves.

Getting Started

Start a simple Vue project with vue-cli and the webpack-simple template.

Then, install `vue-router`.

```
# Yarn
$ yarn add vue-router
# NPM
$ npm install vue-router --save
```

The route configuration we'll be aiming for will be as follows:

```
/ - Home Page
/about - About Page (Really just the home page again...)
 | /nested - A secret nested page
```

Initial Setup

Let's go ahead and get the initial main.js file and configuration ready first. We'll create and modify the other files later. This version won't handle setting the title yet.

main.js

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

```
import Nested from './Nested.vue';

Vue.use(VueRouter);

const routes = [
  {
    path: '/',
    component: App,
    meta: {
      title: 'Home Page - Example App',
      metaTags: [
        {
          name: 'description',
          content: 'The home page of our example app.'
        },
        {
          property: 'og:description',
          content: 'The home page of our example app.'
        }
      ]
    }
  },
  {
    path: '/about',

    component: App,
    meta: {
      title: 'About Page - Example App',
      metaTags: [
        {
          name: 'description',
          content: 'The about page of our example app.'
        },
        {
          property: 'og:description',
          content: 'The about page of our example app.'
        }
      ]
    },

    children: [
      {
        path: 'nested',
        component: Nested,
```

```
      meta: {
        title: 'Nested - About Page - Example App'
      }
    }
  ]
 }
];

const router = new VueRouter({
 routes,
 mode: 'history'
});

new Vue({
 router,
 template: ''
})
.$mount('#app');
```

You'll see that each route has an extra `meta` option, which contains `title`, what will eventually be the route title, and `metaTags`, an object which we will later turn into the page meta tags. (We're not quite there yet.)

The Templates

In the provided `App.vue` file, add a couple of router-links and a router-view for the nested page.

App.vue

```
<template>
  <div id="app">
    <img src="./assets/logo.png">
    <h1>{{msg}}</h1>
    <h2>Essential Links</h2>
    <ul>
      <li><router-link to="/">Home</router-link></li>
      <li><router-link to="/about">About</router-link></li>
      <li><router-link to="/about/nested">Nested Route</router-link></li>
    </ul>
    <h2>Ecosystem</h2>
    <ul>
      <li><a href="http://router.vuejs.org/" target="_blank">vue-router</a></li>
      <li><a href="http://vuex.vuejs.org/" target="_blank">vuex</a></li>
      <li><a href="http://vue-loader.vuejs.org/" target="_blank">vue-
loader</a></li>
      <li><a href="https://github.com/vuejs/awesome-vue"
target="_blank">awesome-vue</a></li>
    </ul>
    <router-view></router-view>
  </div>
</template>
...
```

Now create a new file called `Nested.vue` next to `App.vue`.

Nested.vue

```
<template>
  <p>You've found a hidden alligator nest!</p>
</template>
```

Changing the Title & Meta Tags

Okay, great, if you run `npm run dev` now you should find the app rendering properly, and ought to be able to click the links to navigate routes. Hooray. But wait, the titles and meta tags aren't changing yet.

For that we'll need a custom navigation guard for the router in `main.js`. Let me throw some code at you.

main.js

```
import Vue from 'vue';
import VueRouter from 'vue-router';
```

```
import App from './App.vue';
import Nested from './Nested.vue';

Vue.use(VueRouter);

const routes = [
  {
    path: '/',
    component: App,
    meta: {
      title: 'Home Page - Example App',
      metaTags: [
        {
          name: 'description',
          content: 'The home page of our example app.'
        },
        {
          property: 'og:description',
          content: 'The home page of our example app.'
        }
      ]
    }
  },
  {
    path: '/about',

    component: App,
    meta: {
      title: 'About Page - Example App',
      metaTags: [
        {
          name: 'description',
          content: 'The about page of our example app.'
        },
        {
          property: 'og:description',
          content: 'The about page of our example app.'
        }
      ]
    },

    children: [
      {
        path: 'nested',
```

```
      component: Nested,
      meta: {
        title: 'Nested - About Page - Example App'
      }
    }
  ]
 }
];


router.beforeEach((to, from, next) => {


 const nearestWithTitle = to.matched.slice().reverse().find(r => r.meta &&
r.meta.title);


 const nearestWithMeta = to.matched.slice().reverse().find(r => r.meta &&
r.meta.metaTags);
 const previousNearestWithMeta = from.matched.slice().reverse().find(r => r.meta
&& r.meta.metaTags);


 if(nearestWithTitle) document.title = nearestWithTitle.meta.title;


 Array.from(document.querySelectorAll('[data-vue-router-controlled]')).map(el =>
el.parentNode.removeChild(el));


 if(!nearestWithMeta) return next();


 nearestWithMeta.meta.metaTags.map(tagDef => {
  const tag = document.createElement('meta');

  Object.keys(tagDef).forEach(key => {
   tag.setAttribute(key, tagDef[key]);
  });


  tag.setAttribute('data-vue-router-controlled', '');

  return tag;
```

```
  })

  .forEach(tag => document.head.appendChild(tag));

  next();
});

const router = new VueRouter({
  routes,
  mode: 'history'
});

new Vue({
  router,
  template: ''
})
.$mount('#app');
```

That ought to do the trick. Now when your routes change, the page title will be updated with the closest-matched route's title, and the meta tags will update as well. If you use prerendering, then these changes will be baked into your prerendered HTML files and will work great for SEO. For SSR it can be a bit more difficult. We'll cover that at a later time.

It's also worth noting that dynamic, frequently-updating titles are out of the question with this method. You'll probably have to stick with manually updating `document.title` for such use-cases.