

Curso Técnico Superior Profissional em Tecnologias e Programação de Sistemas de Informação

Unidade Curricular: Desenvolvimento Web – Front-End

1.º Ano/2.º Semestre

Docente: Marco Miguel Olival Olim

Data 23/03/2018

ESTE EXERCÍCIO ABORDA COMPONENTES WEB

No ficheiro index.vue, correspondente à primeira página do site, vamos colocar um cabeçalho (tag header) com uma barra de navegação. São já atribuídas classes para posteriormente aplicar estilos:

```
1 <template>
2   <header class="main-header">
3     <nav class="main-nav">
4       <ul class="nav-links"></ul>
5     </nav>
6   </header>
7 </template>
```

Para efetuar os links poderíamos usar tag <a> como de costume, mas esta força o re-carregamento da página. Para alterar a rota com javascript usamos o vue-router, que também está disponível no NUXT, neste caso concreto a tag nuxt-link que posteriormente será renderizada como e <a> no DOM:

```
1 <template>
2   <header class="main-header">
3     <nav class="main-nav">
4       <ul class="nav-links">
5         <nuxt-link to="/" tag="li"><a>All Posts</a></nuxt-link>
6         <nuxt-link to="/about" tag="li"><a>About</a></nuxt-link>
7       </ul>
8     </nav>
9   </header>
10 </template>
```

Os estilos supracitados, com os links alinhados com flex-box

```
12 <style scoped>
13   .main-header {
14     position: fixed;
15     top: 0;
16     left: 0;
17     width: 100%;
18     background: #022d30;
19     height: 4.5rem;
20   }
21
22   .main-nav {
23     height: 100%;
24   }
25
26   .nav-links {
27     list-style: none;
28     margin: 0;
29     padding: 0;
30     display: flex;
31     justify-content: center;
32     align-items: center;
33     height: 100%;
34   }
```

Cofinanciado por:

Em cada link adicionamos a classe `.nav-link` e retiramos os estilos nativos aos hyperlinks:

```
36 .nav-link {
37   height: 100%;
38   display: flex;
39   justify-content: center;
40   align-items: center;
41   margin: 0 1rem;
42   padding: 0.3rem;
43 }
44
45 .nav-link a {
46   display: block;
47   text-decoration: none;
48   color: white;
49 }
```

Para os estados ativos (e hover) usaremos o azul `#06c4d1`

```
51 .nav-link a:hover,
52 .nav-link a:active,
53 .nav-link.nuxt-link-exact-active a {
54   color: #06c4d1;
55 }
```

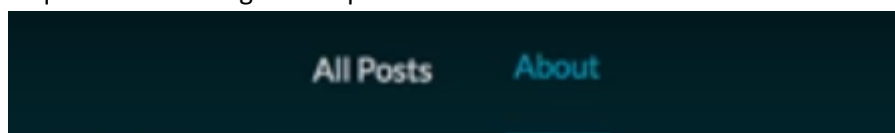
Refira-se que esta classe `nuxt-link-exact-active` é atribuída por default no NUXT e não necessita ser acrescentada aos links

```
4 <ul class="nav-links">
5   <nuxt-link to="/" tag="li" class="nav-link"><a>All Posts</a></nuxt-link>
6   <nuxt-link to="/about" tag="li" class="nav-link"><a>About</a></nuxt-link>
7 </ul>
```

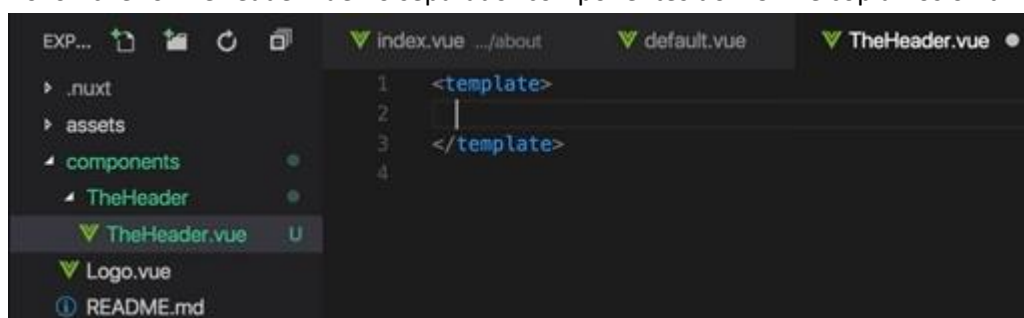
Como último pormenor acrescentamos uma linha para sublinhar o link ativo

```
45 .nav-link.nuxt-link-exact-active {
46   border-bottom: 3px solid #06c4d1;
47 }
```

O que resulta no seguinte aspeto:

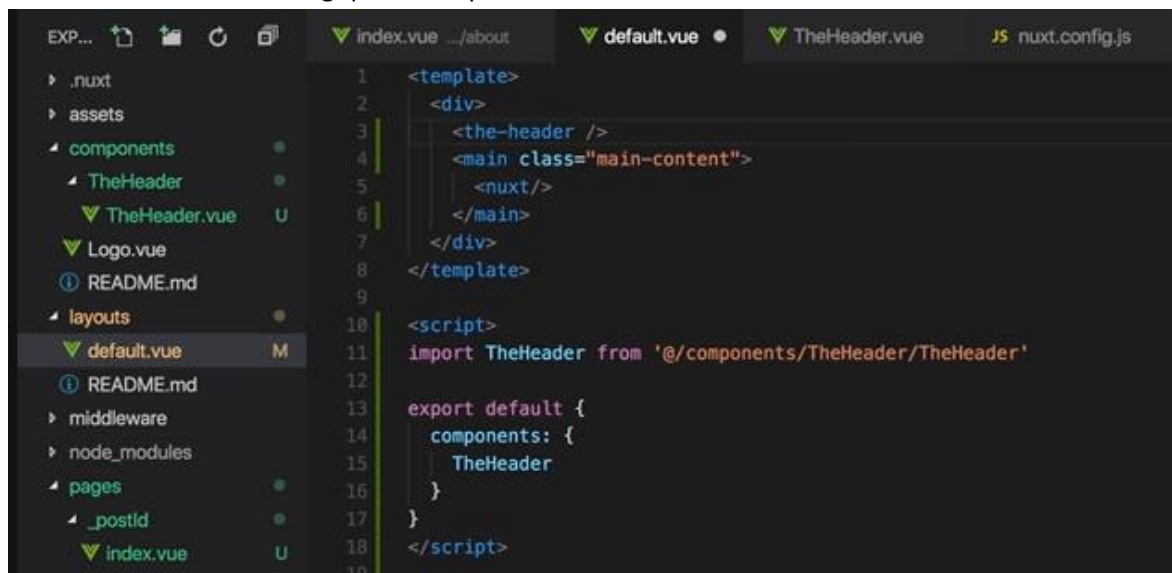


Como esta barra será usada em todas as páginas, teríamos agora de copiar o html para estas e colocar o CSS num ficheiro comum. Uma outra alternativa é converter esta barra para componente. Para tal criamos um novo ficheiro `TheHeader.vue` no separador componentes do NUXT e copiamos o html e CSS correspondente:



Cofinanciado por:

Como pode observar, não há diferenças na estrutura deste ficheiro .vue para os outros que forma criados em pages. Os componentes são meros blocos reutilizáveis. Para utilizá-los numa página basta importá-los e referenciá-los com uma tag que corresponderá ao nome do ficheiro .vue



```
1 <template>
2   <div>
3     <the-header />
4     <main class="main-content">
5       <nuxt/>
6     </main>
7   </div>
8 </template>
9
10 <script>
11 import TheHeader from '@components/TheHeader/TheHeader'
12
13 export default {
14   components: {
15     TheHeader
16   }
17 }
18 </script>
19
```

Como esta barra estará presente em todas as páginas, posso importá-la para todas estas ou para o layout (default.vue) que, como já foi referido, funciona como uma master-page (na tag <nuxt /> será renderizado o restante conteúdo). Quanto aos estilos deste, saliente-se a margin-top com a mesma dimensão da barra para obter aquele espaço atendendo que usamos fixed nesta:



```
21 <style>
22 * {
23   box-sizing: border-box;
24 }
25
26 body {
27   font-family: "Lato", sans-serif;
28 }
29
30 .main-content {
31   margin-top: 4.5rem;
32 }
33 </style>
```

Vamos adicionar mais dois componentes: o hambúrguer, para écrans pequenos, acompanhado de uma barra lateral que irá substituir a barra de navegação nestas situações de dispositivos móveis. Começamos pelo botão que designamos por **TheSideNavToggle** (segundo o guia de estilos do VueJS). Criamos este ficheiro .vue em componentes outra vez:



```
1 <template>
2   <div class="drawer-toggle" role="button">
3     <div class="bar"></div>
4     <div class="bar"></div>
5     <div class="bar"></div>
6   </div>
7 </template>
```

Cofinanciado por:

Os estilos correspondentes já incluem os media queries para que este só apareça em dispositivos pequenos:

```
9 <style scoped>
10 .drawer-toggle {
11   display: flex;
12   flex-direction: column;
13   justify-content: space-around;
14   height: 25px;
15   width: 25px;
16   cursor: pointer;
17 }
18
19 @media (min-width: 768px) {
20   .drawer-toggle {
21     display: none;
22   }
23 }
24
25 .drawer-toggle .bar {
26   width: 90%;
27   height: 2px;
28   background-color: #35495e;
29 }
30 </style>
```

Temos agora de emitir uma ação quando este botão for clicado. Em componentes usamos o `$emit` para esse efeito, que neste caso emite uma string com o valor `toggle`:

```
1 <template>
2   <div
3     class="drawer-toggle"
4     role="button"
5     @click="$emit('toggle')">
6     <div class="bar"></div>
7     <div class="bar"></div>
8     <div class="bar"></div>
9   </div>
10 </template>
```

Vamos adicionar este botão ao componente `TheHeader`. Importamos em componentes e usamos a tag `<TheSideNavToggle>` na template, emitindo a string `sidenavToggle` quando for clicada

```
hjs  ▼ TheSidenav.vue  ▼ TheSideNavToggle.vue  ▼ TheHeader.vue x  JS nuxt.conf
27
28   </div>
29   <TheSideNavToggle @toggle="$emit('sidenavToggle')" />
30 </header>
31 </div>
32 </template>
33
34 <script>
35 import Logo from "~/components/Logo.vue";
36 import TheSideNavToggle from "@components/Navigation/TheSideNavToggle";
37
38 export default {
39   name: "TheHeader",
40   components: {
41     Logo,
42     TheSideNavToggle
43   },
```

Cofinanciado por:



Criamos a barra lateral nos componentes com o nome **TheSidenav.vue**. Esta barra lateral também emite a string *close* quando esta for clicada para ser fechada:

```
h.js  TheSidenav.vue  TheSideNavToggle.vue  TheHeader.vue  JS nuxt.config.js
1  <template>
2  <div class="sidenav-container">
3  |   <div class="sidenav-backdrop" @click="$emit('close')"></div>
4  |   <transition name="slide-side">
5  |     <div class="sidenav">
6  |       <ul class="nav-list" @click="$emit('close')">
7  |         <li class="nav-item">...</li>
8  |         <li class="nav-item">...</li>
9  |         <li class="nav-item">...</li>
10 |       </ul>
11 |     </div>
12 |   </transition>
13 </div>
14 </template>
```

Dentro dos estilos vamos adicionar também uma animação de abertura (na linha 4 já está a tag <transition> com o nome *slide-side* para esta transição):

```
54  .slide-side-enter-active,
55  .slide-side-leave-active {
56  |   transition: all 0.3s ease-out;
57  | }
58  .slide-side-enter,
59  .slide-side-leave-to {
60  |   transform: translateX(-100%);
61  | }
```

Cofinanciado por:



Seguem-se os restantes estilos correspondentes, também com o respetivo media querie:

```
29  <style scoped>
30  .sidenav-container {
31    height: 100%;
32    width: 100%;
33  }
34  .sidenav-backdrop {
35    width: 100%;
36    height: 100%;
37    background-color: rgba(0, 0, 0, 0.7);
38    z-index: 1000;
39    position: fixed;
40    top: 0;
41    left: 0;
42  }
43  .sidenav {
44    height: 100%;
45    width: 300px;
46    background-color: white;
47    z-index: 10000;
48    position: fixed;
49    top: 0;
50    left: 0;
51    box-sizing: border-box;
52    padding: 30px;
53  }
54  .nav-list {
55    list-style: none;
56    padding: 0;
57    margin: 0;
58  }
59  .nav-item {
60    margin: 20px 0;
61  }
62  .nav-item a {
63    text-decoration: none;
64    color: black;
65    font-size: 1.5rem;
66  }
67  .nav-item a:hover,
68  .nav-item a:active,
69  .nav-item a.nuxt-link-active {
70    color: #41B883;
71  }
72  </style>
```

Cofinanciado por:



Já emitimos close quando a barra está fechada mas precisamos agora de receber um valor quando algum botão de fora manda fechar esta barra. Para receber valores os componentes usam a propriedade props. Neste caso a barra irá receber um valor booleano em *show*. (que por default está com o valor false, ou seja, a barra fechada)

```
23 <script>
24 export default {
25   name: "TheSidenav",
26   props: {
27     show: {
28       type: Boolean,
29       default: false
30     }
31   }
32 };
33 </script>
```

Agora já podemos usar um v-if na template para reagir ao valor de *show*

```
▼ TheSidenav.vue x ▼ TheSideNavToggle.vue ▼ TheHeader.vue
<template>
<div class="sidenav-container">
  <div
    v-if="show"
    class="sidenav-backdrop"
    @click="$emit('close')"></div>
  <transition name="slide-side">
    <div
      v-if="show"
      class="sidenav">
        <ul
          class="nav-list"
          @click="$emit('close')">
```

De volta ao layout default, temos de adicionar a nossa barra lateral

```
NavToggle.vue ▼ TheHeader.vue JS nuxt.config.js JS index.js ▼ AdminPostForm.vue ▼ default.vue ●
1 <template>
2   <div>
3     <TheHeader @sidenavToggle="displaySidenav = !displaySidenav" />
4     <TheSidenav :show="displaySidenav" @close="displaySidenav=false"/>
5     <nuxt/>
6   </div>
7 </template>
8 <script>
9   // import navbar from "~/components/navbar.vue";
10  import TheHeader from "~/components/Navigation/TheHeader.vue";
11  import TheSidenav from "@components/Navigation/TheSidenav"
12
13  export default {
14    components: {
15      TheHeader,
16      TheSidenav
17    },
18    data() {
19      return{
20        displaySidenav: false
21      }
22    },
```

Cofinanciado por:

Vamos agora adicionar um cartão para colocar o nosso conteúdo. Criamos o componente `app-quote.vue` da mesma forma que os anteriores. Como este cartão terá um conteúdo genérico, usamos a propriedade `slot` para este ser renderizado:

```
1 <template>
2   <div>
3     <slot></slot>
4   </div>
5 </template>
```

Usamos os seguintes estilos neste componente:

```
div {
  background-color: white;
  box-shadow: 0 0 24px rgba(0, 0, 0, 0.3);
  padding: 3rem;
  width: 420px;
  position: relative;
  overflow: hidden;
  display: flex;
  flex-direction: column;
  justify-content: space-between;
}
```

Ao ser importado este componente será usado desta forma:

```
1 <template>
2   <div class="container">
3     <div class="row">
4       <div class="col-xs-12">
5         <app-quote>
6           <h2>The Quote</h2>
7           <p>A wonderful Quote</p>
8         </app-quote>
9       </div>
10    </div>
11  </div>
12 </template>
13
14 <script>
15   import Quote from './components/Quote.vue';
16
17   export default {
18     components: {
19       appQuote: Quote
20     }
21   }
22 </script>
23
24 <style>
```

Cofinanciado por: