Vue项目使用CSS变量实现主题化

倔强的小石头 发布于 2019-11-23

主题化管理经常能在网站上看到，一般的思路都是将主题相关的CSS样式独立出来，在用户选择主题的时候加载相应的CSS样式文件。现在大部分浏览器都能很好的兼容CSS变量，主题化样式更容易管理了。最近，使用CSS变量在Vue项目中做了一个主题化实践，下面来看看整个过程。

Github项目地址
演示地址

可行性测试

为了检验方法的可行性，在public文件夹下新建一个themes文件夹，并在themes文件夹新建一个default.css文件：

```
:root {
  --color: red;
}
```

在public文件夹的index.html文件中引入外部样式theme.css，如下：

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,initial-scale=1.0">
    <link rel="icon" href="<%= BASE_URL %>favicon.ico">
    <title>vue-skin-peeler-demo</title>
    <!-- 引入themes文件夹下的default.css -->
    <link rel="stylesheet" type="text/css" href="src/themes/default.css" rel="e
  </head>
  <body>
    <noscript>
      <strong>We're sorry but vue-skin-peeler-demo doesn't work properly without
    </noscript>
    <div id="app"></div>
    <!-- built files will be auto injected -->
```

然后，在Home.vue中使用CSS变量：

```
<template>
  <div class="home">
    <div :class="$style.demo">变红色</div>
  </div>
</template>

<script>
export default {
  name: 'home'
}
</script>

<style module lang="scss">
  .demo {
    color: var(--color);
  }
</style>
```

然后，运行项目并在浏览器中打开页面，页面显示效果正常。

> 注意：@vue/cli使用link标签引入css样式可能报错“We're sorry but vue-skin-peeler-demo doesn't work properly without JavaScript enabled. Please enable it to continue.”。这是因为@vue/cli将src目录下的文件都通过webpack打包所引起，所以，静态文件资源要放在public（如果是@vue/cli 2.x版本放在static）文件夹下。

实现主题切换

这里主题切换的思路是替换link标签的href属性，因此，需要写一个替换函数，在src目录下新建themes.js文件，代码如下：

```
// themes.js
const createLink = (() => {
  let $link = null
  return () => {
    if ($link) {
      return $link
    }
    $link = document.createElement('link')
    $link.rel = 'stylesheet'
    $link.type = 'text/css'
```

```
  }
})()

/**
 * 主题切换函数
 * @param {string} theme - 主题名称, 默认default
 * @return {string} 主题名称
 */
const toggleTheme = (theme = 'default') => {
  const $link = createLink()
  $link.href = `./themes/${theme}.css`
  return theme
}

export default toggleTheme
```

然后，在themes文件下创建default.css和dark.css两个主题文件。创建CSS变量，实现主题化。CSS变量实现主题切换请参考另一篇文章初次接触css变量

兼容性

IE浏览器以及一些旧版浏览器不支持CSS变量，因此，需要使用css-vars-ponyfill，是一个ponyfill，可在旧版和现代浏览器中为CSS自定义属性（也称为“ CSS变量”）提供客户端支持。由于要开启watch监听，所以还有安装MutationObserver.js。

安装：

```
npm install css-vars-ponyfill mutationobserver-shim --save
```

然后，在themes.js文件中引入并使用：

```
// themes.js
import 'mutationobserver-shim'
import cssVars from 'css-vars-ponyfill'

cssVars({
  watch: true
})

const createLink = (() => {
  let $link = null
```

```
      return $link
    }
    $link = document.createElement('link')
    $link.rel = 'stylesheet'
    $link.type = 'text/css'
    document.querySelector('head').appendChild($link)
    return $link
  }
})()

/**
 * 主题切换函数
 * @param {string} theme - 主题名称, 默认default
 * @return {string} 主题名称
 */
const toggleTheme = (theme = 'default') => {
  const $link = createLink()
  $link.href = `./themes/${theme}.css`
  return theme
}

export default toggleTheme
```

开启watch后，在IE 11浏览器点击切换主题开关不起作用。因此，每次切换主题时都重新执行cssVars()，还是无法切换主题，原因是开启watch后重新执行cssVars()是无效的。最后，只能先关闭watch再重新开启。成功切换主题的themes.js代码如下：

```
// themes.js
import 'mutationobserver-shim'
import cssVars from 'css-vars-ponyfill'

const createLink = (() => {
  let $link = null
  return () => {
    if ($link) {
      return $link
    }
    $link = document.createElement('link')
    $link.rel = 'stylesheet'
    $link.type = 'text/css'
    document.querySelector('head').appendChild($link)
    return $link
  }
})()
```

```
   * @param {string} theme - 主题名称, 默认default
   * @return {string} 主题名称
   */
  const toggleTheme = (theme = 'default') => {
    const $link = createLink()
    $link.href = `./themes/${theme}.css`
    cssVars({
      watch: false
    })
    setTimeout(function () {
      cssVars({
        watch: true
      })
    }, 0)
    return theme
  }

  export default toggleTheme
```

查看所有代码，请移步Github项目地址。

记住主题

实现记住主题这个功能，一是可以向服务器保存主题，一是使用本地存储主题。为了方便，这里主要使用本地存储主题的方式，即使用localStorage存储主题。具体实现请移步Github项目地址。

阅读 4.5k · 更新于 2019-11-23