vue-meta让你更优雅的管理头部标签

原创 2018-01-17 yingye 前端新视野

本文摘要

来自摩拜前端团队 yingye ，如有不对地方请指正

本文主要介绍了 vue-meta 的使用和源码分析，文中如有任何表述不清或不当的地方，欢迎大家批评指正。

欢迎关注本系列，留言分享 ssr 的一些经验

在 Vue SPA 应用中，如果想要修改 HTML 的头部标签，或许，你会在代码里，直接这么做:

```
// 改下title
document.title = 'what?'

// 引入一段script
let s = document.createElement('script')
s.setAttribute('src', './vconsole.js')
document.head.appendChild(s)
// 修改meta信息，或者给html标签添加属性...
// 此处省略一大坨代码...
```

今天给大家介绍一种更优雅的方式，去管理头部标签 `vue-meta`

vue-meta

> Manage page meta info in Vue 2.0 components.
>
> SSR + Streaming supported. Inspired by react-helmet.

借用 vue-meta github 上的介绍，基于 Vue 2.0 的 vue-meta 插件：

主要用于管理 HMTL 头部标签，同时也支持 SSR。

vue-meta有以下特点：

- 在组件内设置 `metaInfo`，便可轻松实现头部标签的管理
- `metaInfo` 的数据都是响应的，如果数据变化，头部信息会自动更新
- 支持 SSR

如何使用

在介绍如何使用之前，先和大家普及一个最近很火的名词 **服务端渲染（SSR, Server Side Render）**，简单来讲，就是在访问某个页面时，服务端会把渲染好的页面，直接返回给浏览器。

我们知道 vue-meta 是支持 SSR 的，下面的介绍分成两部分：

Client 客户端

在入口文件中，install vue-meta plugin

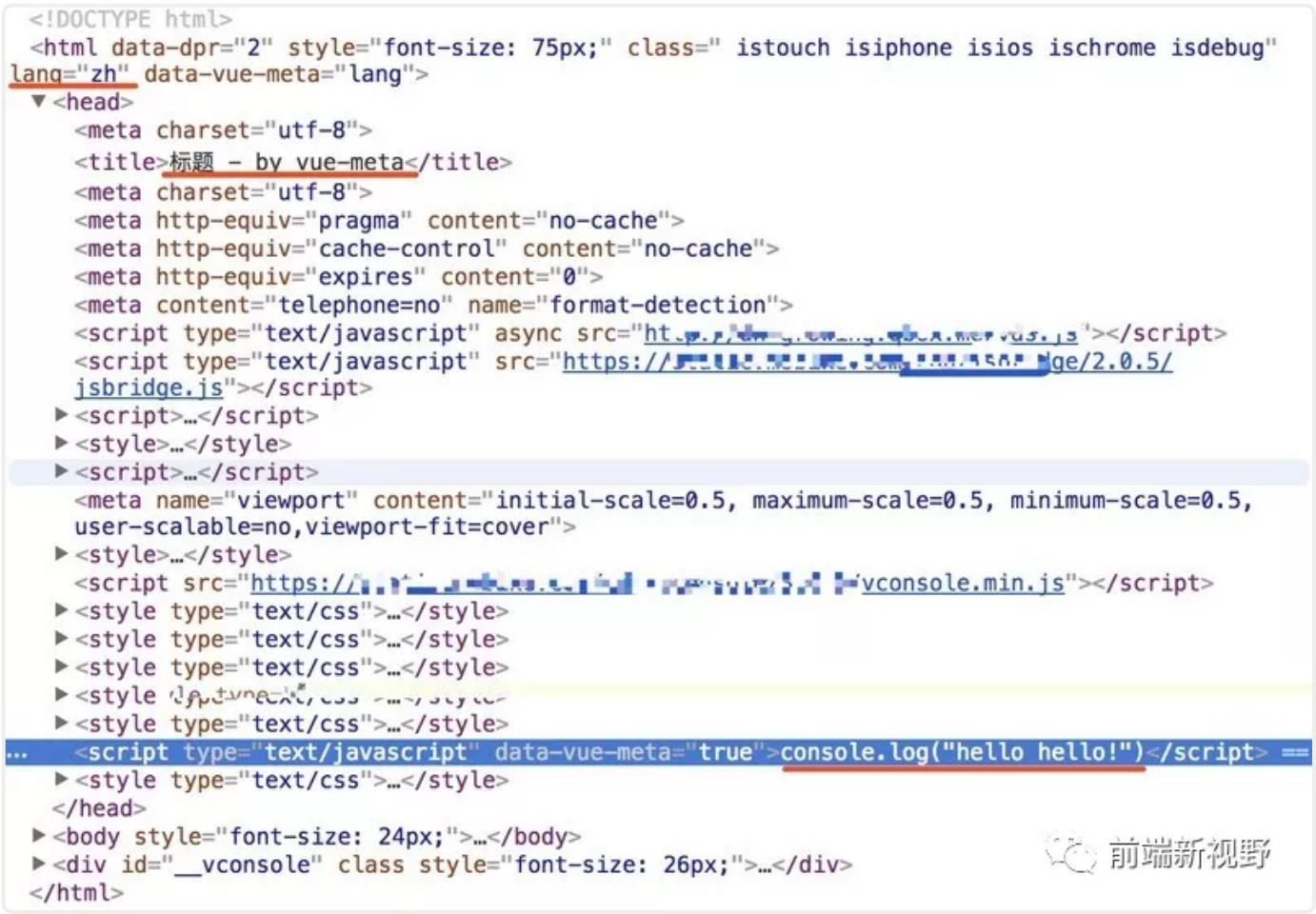
```
import Vue from 'vue'
import VueRouter from 'vue-router'

import VueMeta from 'vue-meta'


Vue.use(VueRouter)
Vue.use(VueMeta)


/* eslint-disable no-new */
new Vue({
  el: '#app',
  router,
  template: '<App/>',
  components: { App }
})
```

然后就可以在组件中使用了

```
export default {
  data () {
    return {
      myTitle: '标题'
    }
  },
  metaInfo: {
    title: this.myTitle,
    titleTemplate: '%s - by vue-meta',
    htmlAttrs: {
      lang: 'zh'
    },
    script: [{innerHTML: 'console.log("hello hello!")', type: 'text/javascript'}],
```

```
      __dangerouslyDisableSanitizers: ['script']
    },
    ...
  }
```

可以看一下页面显示



熟悉 Nuxt.js 的同学，会发现配置 meta info 的 keyName 不一致。可以通过下面的配置方法来修改：

```
// vue-meta configuration
Vue.use(Meta, {
  keyName: 'head', // the component option name that vue-meta looks for meta info on.
  attribute: 'data-n-head', // the attribute name vue-meta adds to the tags it observes
  ssrAttribute: 'data-n-head-ssr', // the attribute name that lets vue-meta know that m
  tagIDKeyName: 'hid' // the property name that vue-meta uses to determine whether to o
})
```

更加全面详细的 api，可以参考 vue-meta github

Server 服务端

Step 1. 将 $meta 对象注入到上下文中

server-entry.js:

```
import app from './app'

const router = app.$router
const meta = app.$meta() // here

export default (context) => {
  router.push(context.url)
  context.meta = meta // and here
  return app
}
```

`$meta` 主要提供了： `inject` 和 `refresh` 方法

`inject` 方法，用在服务端，返回设置的metaInfo

`refresh` 方法，用在客户端，作用是更新meta信息。

Step 2. 使用 inject() 方法 输出页面

server.js:

```
app.get('*', (req, res) => {
  const context = { url: req.url }
  renderer.renderToString(context, (error, html) => {
    if (error) return res.send(error.stack)
    const bodyOpt = { body: true }
    const {
      title, htmlAttrs, bodyAttrs, link, style, script, noscript, meta
    } = context.meta.inject()
    return res.send(`
      <!doctype html>
      <html data-vue-meta-server-rendered ${htmlAttrs.text()}>
        <head>
          ${meta.text()}
          ${title.text()}
          ${link.text()}
          ${style.text()}
          ${script.text()}
          ${noscript.text()}
        </head>
        <body ${bodyAttrs.text()}>
          ${html}
          <script src="/assets/vendor.bundle.js"></script>
          <script src="/assets/client.bundle.js"></script>
```

```
            ${script.text(bodyOpt)}
          </body>
        </html>
      `)
    })
  })
```

源码分析

前面说了 vue-meta 的使用方法，或许大家会想这些功能是怎么实现的，那下面就和大家分享一下源码。

怎么区分 client 和 server渲染?

vue-meta 会在 `beforeCreate()` 钩子函数中，将组件中设置的 metaInfo ，放在 this.$metaInfo 中。我们可以在其他生命周期中，访问 this.$metaInfo 下的属性。

```
if (typeof this.$options[options.keyName] === 'function') {
  if (typeof this.$options.computed === 'undefined') {
    this.$options.computed = {}
  }
  this.$options.computed.$metaInfo = this.$options[options.keyName]
}
```

vue-meta 会在 created 等生命周期的钩子函数中，监听 `$metaInfo` 的变化，如果发生改变，就调用 `$meta` 下的 `refresh` 方法。

这也是 `metaInfo` 做到响应的原因。

```
created () {
  if (!this.$isServer && this.$metaInfo) {
    this.$watch('$metaInfo', () => {
      batchID = batchUpdate(batchID, () => this.$meta().refresh())
    })
  }
},
```

Server 端，主要是暴露 `$meta` 下的 `inject` 方法，调用 `inject` 方法，会返回对应的信息。

client 和 server 端 是如何修改标签的?

client端修改标签，就是本文开头提到的：通过原生 js，直接修改

```
  return function updateTitle (title = document.title) {
    document.title = title
  }
```

server 端，就是通过 text 方法，返回 string 格式的标签

```
  return function titleGenerator (type, data) {
    return {
      text () {
        return `<${type} ${attribute}="true">${data}</${type}>`
      }
    }
  }
```

__dangerouslyDisableSanitizers 做了什么?

vue-meta 默认会对特殊字符串进行转义，如果设置了 `__dangerouslyDisableSanitizers`，就不会对再做转义处理。

```
  const escapeHTML = (str) => typeof window === 'undefined'
    // server-side escape sequence
    ? String(str)
      .replace(/&/g, '&amp;')
      .replace(/</g, '&lt;')
      .replace(/>/g, '&gt;')
      .replace(/"/g, '&quot;')
      .replace(/'/g, '&#x27;')
    // client-side escape sequence
    : String(str)
      .replace(/&/g, '\u0026')
      .replace(/</g, '\u003c')
      .replace(/>/g, '\u003e')
      .replace(/"/g, '\u0022')
      .replace(/'/g, '\u0027')
```

最后

最开始接触 vue-meta 是在 Nuxt.js 中。如果想了解 Nuxt.js，欢迎大家阅读我们之前的推送文章 Nuxt.js 踩坑分享。文中有任何表述不清或不当的地方，欢迎大家批评指正。

昨日的日刊是不是漏看了，给你一个传送门：

前端日刊 - 1/16

Nuxt.js 系列文章：

[Nuxt.js 系列] 踩坑分享篇

好消息：

1、可以在公众号【菜单 - 日刊】上看到所有的日刊文章模板啦

2、日刊君也开通了知乎专栏 [前端新视野]：
https://zhuanlan.zhihu.com/c_141430263

一个一天就破 *k 关注的日刊号
一个立志把质量当生命的日刊号
一个大佬们都在关注的日刊号
...

分享给喜欢学习的小伙伴吧

阅读原文