Vue SSR

> Vue Server Side Render

SPA 单页面应用程序有个最大的缺点：不利于 SEO。

如何解决？服务端渲染，在服务端完成对页面的解析替换，发送给浏览器的直接就是结果。

1. 介绍

- SSR（服务端渲染）：服务器直接生成HTML文档并返回给浏览器，但页面交互能力有限。适用于任何后端语言：PHP、Java、Python、GO等。
  - 好处：响应速度快，有利于 SEO
  - 缺点：前后端代码混在一起，难以开发和维护，不适合进行前后端分离开发
- CSR（客户端渲染）：页面初始加载的HTML文档中无内容，需要下载执行JS文件，由浏览器动态生成页面，并通过JS进行页面交互事件与状态管理。
  - 好处：尤其适合前后端分离开发，好开发，好维护，单页面应用中几乎都是客户端渲染
  - 缺点：首次加载缓慢，不利于 SEO（几乎为0）
  - 注意：CSR 不一定是 SPA
- SPA（单页面应用程序）
  - 好处：页面导航不用刷新整个页面，体验好，有利于前后端分离开发
  - 缺点：不利于 SEO（因为单页面应用中都是使用客户端渲染的方式），还有首次响应慢（第1次要加载大量的公共资源）
- isomorphic web apps（同构应用）：isomorphic/universal，基于react、vue框架，客户端渲染和服务器端渲染的结合，在服务器端执行一次，用于实现服务器端渲染（首屏直出），在客户端再执行一次，用于接管页面交互，核心解决SEO和首屏渲染慢的问题。

我们常说的渲染指的是什么：说白了就是解析替换 HTML 模板字符串。

1.1. 什么是CSR（客户端渲染）

> CSR: Client Side Render

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
</head>
<body>
  <div id="app">
    <h1>{{ message }}</h1>
    <ul>
      <li v-for="item in todos" :key="item.id">{{ item.title }}</li>
    </ul>
  </div>
```

```
  <!-- csr: client side render -->
  <script src="https://cdn.jsdelivr.net/npm/vue@2.6.10/dist/vue.js"></script>
  <script>
    new Vue({
      el: '#app',
      data: {
        message: 'hello world',
        todos: []
      },

      created () {
        setTimeout(() => {
          this.todos = [
            { id: 1, title: '吃饭' },
            { id: 2, title: '睡觉' },
            { id: 3, title: '打豆豆' }
          ]
        }, 500)
      }
    })
  </script>
</body>
</html>
```

总结：

- 服务端返回一个空的页面结构
- 客户端加载执行对应的 JavaScript 脚本
- 客户端操作 DOM 完成解析替换

1.2. 什么是SSR（服务端渲染）

> SSR: Server Side Render

安装依赖：

```
# 创建 http 服务
npm i express

# 服务端模板引擎
npm i art-template express-art-template
```

服务端代码：

```
const express = require('express')

const app = express()

app.engine('html', require('express-art-template'))

app.get('/', (req, res) => {
  res.render('index.html', {
    message: 'Hello World',
    todos: [
```

```
      { id: 1, title: '吃饭' },
      { id: 2, title: '睡觉' },
      { id: 3, title: '打豆豆' }
    ]
  })
})

app.listen(3000, () => console.log('Server running at port 3000.'))
```

客户端代码：

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
</head>
<body>
  <h1>{{ message }}</h1>
  <ul>
    {{ each todos }}
    <li>{{ $value.title }}</li>
    {{ /each }}
  </ul>
</body>
</html>
```

总结：

- 页面在服务端进行渲染（解析替换）
- 然后把渲染的结果发送给客户端
- 客户端只负责展示就可以了

1.3. 什么是 SEO

SEO（Search Engine Optimization）：汉译为搜索引擎优化。是一种方式：利用搜索引擎的规则提高网站在有关搜索引擎内的自然排名。目的是让其在行业内占据领先地位，获得品牌收益。很大程度上是网站经营者的一种商业行为，将自己或自己公司的排名前移。

1.4. 为什么 CSR 不利于 SEO

使用程序获取网页内容，根据一套特定规则分析网页内容。

使用程序获取并分析网页内容的行为，称之为爬虫。

- 如何使用程序获取网页内容
- 分析内容数据收录

获取网页内容：

```
/**
```

```
 * 任何服务端技术都可以做爬虫
 * Java、PHP、Python、Ruby、。。。。Node.js。。。
 */
const http = require('http')

// 1. 获取网页内容
http.get('http://127.0.0.1:3000/', (res) => {

// http.get('http://127.0.0.1:5500/csr.html', (res) => {
  let rawData = '';
  res.on('data', (chunk) => { rawData += chunk; });

  res.on('end', () => {
    console.log(rawData)
  })
})


// 2. 分析
```

1.5. 总结

- 客户端渲染
  - 不利于 SEO
- 服务端渲染
  - 更好的 SEO
  - 更快的内容到达时间
- SPA
  - 单页面应用程序
  - 单页面应用程序中都是客户端渲染出来的，就不要提什么 SEO 了
  - 单页面首次访问比较缓存

注意：如果没有 SEO 等需求，就不要搞服务端渲染，反而麻烦了。

服务端渲染需要具备更高的综合能力：

- Node
- webpack
- 。。。。

2. Vue SSR

Vue SSR 充当的就是上面示例中的 art-template 的角色。

说白了我们可以在服务端使用 Vue，和客户端中的 Vue 稍有区别。

服务端中的 Vue 充当的就是模板引擎的角色。

3. Nuxt