

第五章 服务端渲染

一样的在线教育，不一样的教学品质



目录 Contents

- ◆ 服务端渲染简介
- ◆ 实现React服务端渲染
- ◆ 路由的处理
- ◆ Redux应用
- ◆ 样式处理
- ◆ Next.js服务端渲染框架应用

1. 服务端渲染简介

什么是服务端渲染

前端渲染

html页面作为静态文件存在，前端请求时后端不对该文件做任何内容上的修改，直接以资源的方式返回给前端，前端拿到页面后，根据写在html页面上的js代码，对该html的内容进行修改。

服务端渲染

前端发出请求后，后端在将HTML页面返回给前端之前，先把HTML页面中的特定区域，用数据填充好，再将完整的HTML返回给前端。（在浏览器端查看源代码可以看到文本内容）在SPA场景下，服务端渲染都是针对第一次get请求，它会完整的html给浏览器，浏览器直接渲染出首屏，用不着浏览器端多一个AJAX请求去获取数据再渲染。

1. 服务端渲染简介

前端渲染优势与劣势



优势

前后端分离，前端专注UI，
后端专注api开发



劣势

前端渲染首屏（第一次看到页面上内容的时间）渲染慢
不利于SEO（搜索引擎优化）的优化

1. 服务端渲染简介

服务端渲染优势与劣势



优势

有利于SEO优化
首屏渲染快



劣势

更多的服务器端负载
不容易维护

如何选择?

建议:

- 如果注重SEO的新闻站点，非强交互的页面，建议用SSR（server side render）；
- 像后台管理页面这类强交互的应用，建议使用前端渲染。



目录 Contents

- ◆ 服务端渲染简介
- ◆ 实现React服务端渲染
- ◆ 路由的处理
- ◆ Redux应用
- ◆ 样式处理
- ◆ Next.js服务端渲染框架应用

1.实现React服务端渲染

1.1在服务端构建React应用

安装express框架: `npm install express --save`

通过node编写一个简单的服务端程序

```
var express=require('express')
var app=express();
app.get('/',function(req,res){
  res.send(`
    <html>
      <head>
        <title>Hello World</title>
        <body>Hello World</body>
      </head>
    </html>
  `)
})
var server=app.listen(3000);
```


■ 1.实现React服务端渲染

react的安装 :`npm install react --save`

react-dom的安装

```
npm install react-dom
```

```
npm install --save-dev @babel/preset-react
```

1.实现React服务端渲染

1.2webpack配置

安装webpack

```
npm install --save-dev webpack  
npm install --save-dev webpack-cli
```

安装babel-loader

```
npm install -D babel-loader @babel/core @babel/preset-env webpack
```

1.实现React服务端渲染

1.3自动打包与自动重启

配置自动打包

```
"scripts": {  
  "start": "node ./build/bundle.js",  
  "build": "webpack --config webpack.server.js --watch"  
}
```

自动启动服务器

```
npm install nodemon -g
```

```
"scripts": {  
  "start": "nodemon --watch build --exec node ./build/bundle.js",  
  "build": "webpack --config webpack.server.js --watch"  
}
```

■ 1.实现React服务端渲染

1.4同构处理

何为“**同构**”，简单来说就是“**同种结构的不同表现形态**”。

通俗的说：同一份react代码在服务端执行一遍，再在客户端执行一遍。

■ 1.实现React服务端渲染

1.5webpack合并处理

我们写了两个关于webpack的配置文件，一个是webpack.client.js，另外一个为webpack.server.js。同时，我们发现这两个文件中，很多的配置项代码是重复的，所以这里可以将重复的内容单独的抽离处理，和这两个文件进行合并的处理。

需要借助于： **webpack-merge** 这款工具来完成

```
npm install webpack-merge
```



一样的在线教育，不一样的教学品质