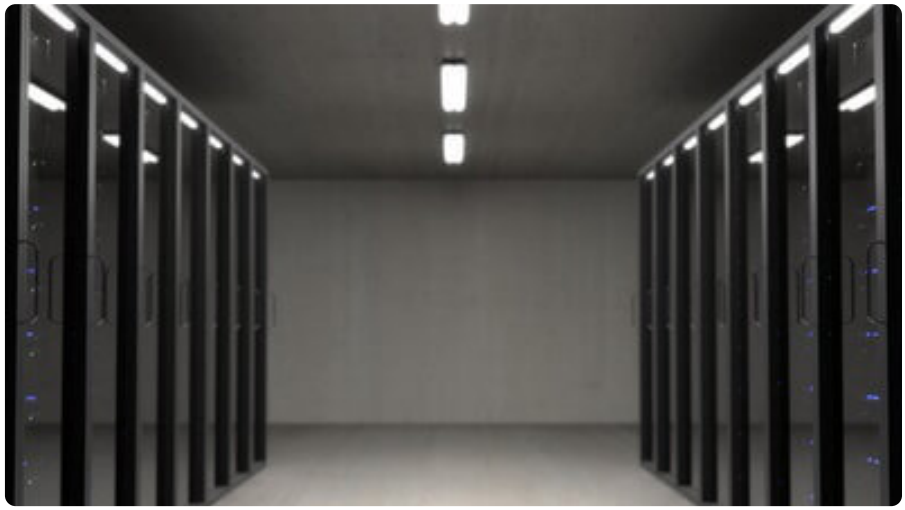
目录

24 服务端渲染

更新时间：2020-06-05 18:57:17



> 耐心是一切聪明才智的基础。
>
> ——柏拉图

引入

服务端渲染在一般场景下其实是无需使用的，虽然它可以解决白屏问题以及SEO优化，但是它对服务端要求非常高，如果公司不是为了极致的用户体验，一般的客户端渲染也完全可以满足日常需要。如此，服务端渲染的优点和缺点大家也大致了解了，下面就来具体介绍服务端渲染。

各种渲染方式

后端渲染

在没有React、Vue框架之前，前端通常都是直接使用原生JavaScript，或者是框架JQuery进行日常开发。当时前端页面比较简单，我们前端把页面开发完成之后，然后交给后端，再然后后端把页面嵌入到对应的后端代码当中，后端直接生成HTML文档并返回给浏览器，页面交互能力有限。这样的开发页面渲染采用的就是后端渲染的形式，比较常用的后端语言(PHP、Java、Python、GO)都支持这种形式。

客户端渲染(CSR)

在React、Vue兴起之后，前后端的工作变得更加明确，协作方式逐渐过渡到了前后端分离的这种形式。这种情况下，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JavaScript进行页面交互事件与状态管理，这样的页面渲染采用的就是客户端渲染。采用这种开发方式我们在部署的时候就可以单独进行部署，无需依赖后端，当前的主流开发方式就是这种方式。

服务端渲染(SSR)

接管页面交互。这样页面渲染采用的就是服务端渲染。

本地离线渲染(NSR)

本地离线渲染这个是我今年参加GMTC大会才接触到的，是一项比较新的技术，具体原理就是通过离线资源和预加载的方式，native提前完成渲染页面并放入缓存，等用户点击的时候瞬间呈现页面。这种技术比较新，目前还在摸索阶段，大家了解即可。

各种渲染方式的区别

因为目前客户端渲染(CSR)和服务端渲染(SSR)是当前的主要渲染方式，所以我们主要来介绍这两者的区别，首先我们来说CSR，先看一张图：

上图就是客户端渲染的整个流程的简单示意，可以看到客户端渲染的TTFP比较长，来回经历了3个HTTP请求周期：首先是加载HTML文档，然后HTML路径再拉取JavaScript文件，紧接着根据JavaScript文件当中Ajax请求拉取对应的数据，最后根据数据渲染页面。这就是客户端渲染的一个大致流程，来回三次的HTTP请求就是导致白屏的主要原因。

接下来我们再来看服务端渲染，服务端渲染需要客户端与服务端的共同配合，关于服务端渲染的大概流程如下图：

如上可以看到，我们第一次请求服务端就已经返回了已经可以渲染的HTML，这样仅需要一次请求就可以进行页面的渲染， 相比前面三次请求大大缩短了时间，这样我们就能够更快看到页面。

如何实现服务端渲染

React 服务端渲染

说了这么多，到底如何实现服务端渲染呢，下面我们就来分别介绍React和Vue如何实现服务端渲染。目前这两个框架都是支持服务端渲染的，首先我们来介绍React如何进行服务端渲染。

目，如下：

```
npm install -g create-next-app  //安装脚手架
create-next-app next_test //创建项目
```

运行如上命令这样我们就创建了一个简单的Next项目，下面我们来看一下package.json文件，如下：

可以看到依赖非常简单，运行命令npm run dev即可进入开发者模式进行开发，运行之后的页面就是默认的Next欢迎页面。因为默认的页面是静态的，并没有从外面获取数据，所以代码也非常简单，如下：

```
import React from 'react';

const Home = () => {
    <h1>
        // 静态内容
    </h1>
}

export default Home
```

上面由于是静态内容，所以相对来说简单一些，但通常业务都是需要从外部调用API来获取数据的，所以引出了我们下面要介绍的getInitialProps这个方法。**如果我们需要发起请求从外部获取数据，那么都要放到getInitialProps这个方法当中**，如下：

```
const Home = (props) => {
        <h1>
                {props.data}
        </h1>
}

Home.getInitialProps = async () => {
      // ajax请求
}
```

如上我们把获取数据的方法放到getInitialProps当中，剩下的工作就可以交给Next去做，最后数据以props的形式传递给相应的组件。以上就是简单的一个服务端渲染的小案例，一些其他的高级用法，大家可以直接阅读官方文档。

Vue服务端渲染

目录


```
npm install -g create-nuxt-app // 安装脚手架
create-nuxt-app nuxt-test // 创建项目
```

在Nuxt当中，同样有用于获取数据的对应API方法asyncData，我们把对应的获取数据的异步操作放到asyncData即可，如下：

```
<template>
  <div>
    {{info}}
  </div>
</template>
<script>
import axios from 'axios'
export default {
  asyncData() {
    return axios.get('api地址')
                .then((res) => {
                   return {info: res.data}
                })
  }
}
</script>
```

以上就是Vue服务端渲染的小案例，同样的如果想更深入的了解关于服务端渲染的内容，可以阅读官方文档。

小结

这一节我们介绍了服务端渲染的概念以及在React和Vue当中如何做服务端渲染。这里介绍的都是开箱即用的工具，当然大家也可以使用原生提供的方法。最后再说一点，服务端渲染的性能虽然好，但是对服务端的压力非常大，所以大家还是要根据实际业务来选用，在性能要求较高的场景下，服务端渲染是个不错的选择。

← 23 让加载“懒”一点

25 移动端的优化技巧也想让你知道 →

精选留言 0

欢迎在这里发表留言，作者筛选后可公开显示

目前暂无任何讨论

目录