SSR 服务端渲染

黄初凤

2020.06.15

一、SSR 是什么

二、为什么使用SSR

三、怎么做

一

SSR 是什么

SSR 是什么

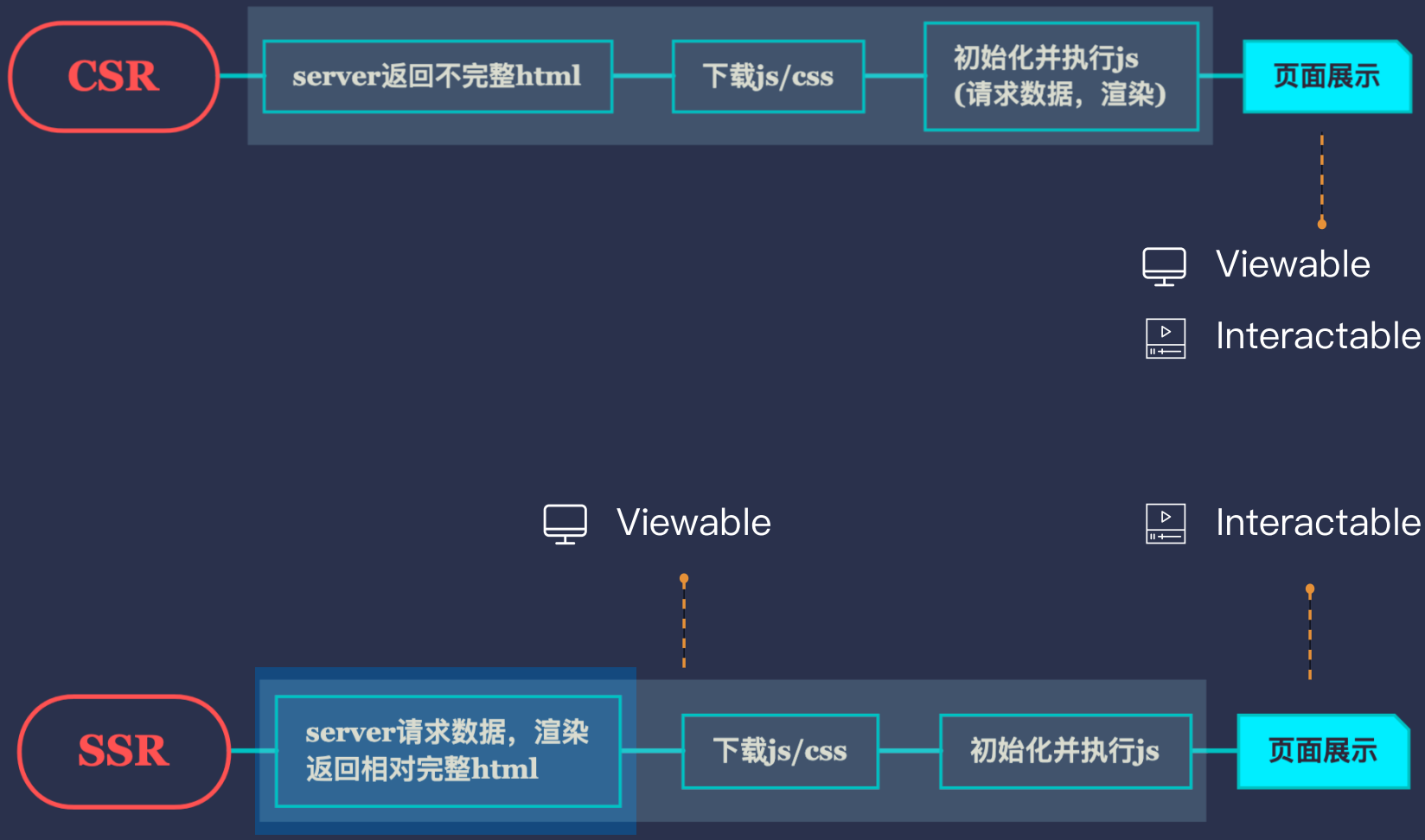
Server Side Rendering

服务器将组件或页面生成html字符串，再返回给客户端，最后将这些静态标记"激活"为客户端上完全可交互的应用程序

所见即所得

页面上呈现的内容，在 html 源文件里也能找到

SSR vs CSR


CSR
server返回不完整html
下载js/css
初始化并执行js
(请求数据，渲染)
页面展示
Viewable
Interactable
Viewable
Interactable
SSR
server请求数据，渲染
返回相对完整html
下载js/css
初始化并执行js
页面展示

二

为什么使用SSR

为什么使用SSR

- 有利于SEO
- 提升在手机和低功耗设备上的性能
- 更利于首屏渲染

– 局限性 ？

SSR 局限性

- 服务端压力大
- 开发条件受限
- 对构建设置和部署需更多要求

– 预渲染（Prerendering）

Prerendering

```
const path = require('path')
const PrerenderSPAPlugin = require('prerender-spa-plugin')

module.exports = {
  plugins: [
    ...
    new PrerenderSPAPlugin({
      // Required - The path to the webpack-outputted app to prerende
      staticDir: path.join(__dirname, 'dist'),
      // Required - Routes to render.
      routes: [ '/', '/about', '/some/deep/nested/route' ],
    })
  ]
```

三

怎么做

需要注意的点

1. `window, document, navigator` 等浏览器对象不能直接使用

   可以通过PLATFORM_ID标记注入的Object来检查当前平台是浏览器还是服务器

```
import { PLATFORM_ID } from '@angular/core';
 import { isPlatformBrowser, isPlatformServer } from '@angular/common';

 constructor(@Inject(PLATFORM_ID) private platformId: Object) { ... }

 ngOnInit() {
   if (isPlatformBrowser(this.platformId)) {
      // only in browser
   }
   if (isPlatformServer(this.platformId)) {
     // only on server
   }
}
```

需要注意的点

2. **不要直接操作nativeElement**

   使用Renderer2，从而可以跨平台改变应用视图

```
ngAfterViewInit() {
    // bad
    this.elRef.nativeElement.style.color = 'red';

    // good
    this.renderer.setStyle(this.elRef.nativeElement, 'color','red');
}
```

SSR 应用场景

- 对SEO有要求
- 交互性不强
- 首屏加载速度

Thanks

黄初凤

2020.06.15