React18 SSR + Next.js 13

刘军　liujun

目录
content

1 邂逅 Next.js

2 Next.js初体验

3 全局配置

4 内置组件

5 样式和资源

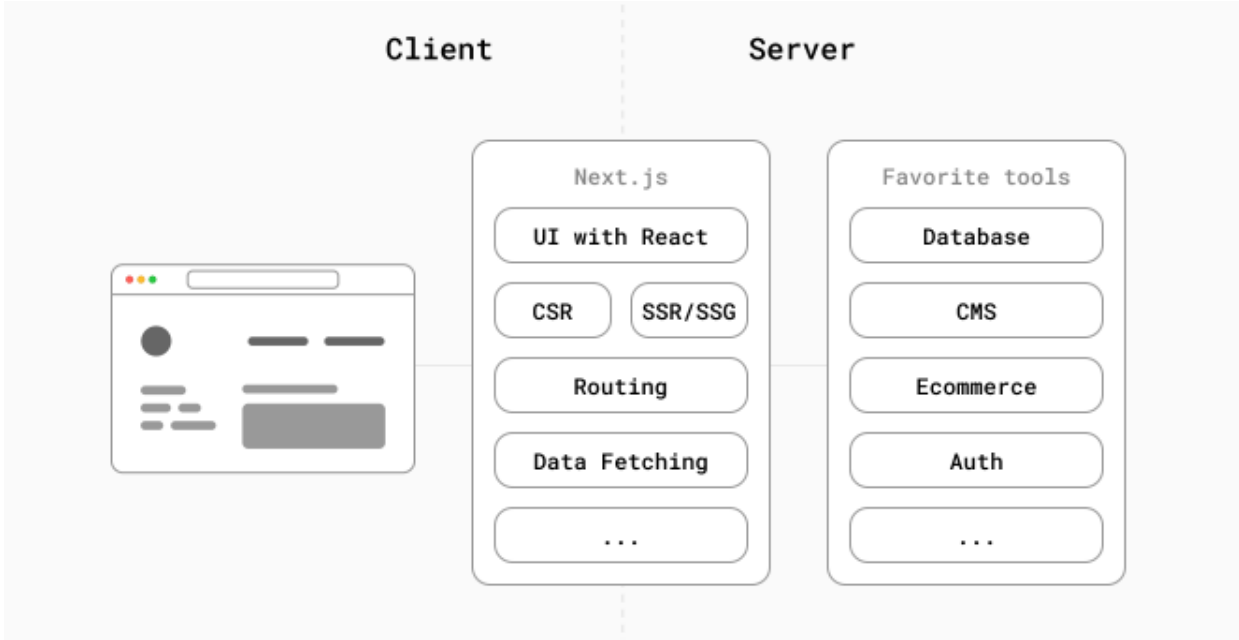
6 页面和导航

7 动态路由

什么是Next?

- Next.js 是一个**React框架**，支持CSR、SSR、SSG、ISR (Incremental Static Regeneration)等渲染模式。
- Next.js 提供了创建 Web 应用程序的构建块，比如：
  - 用户界面、路由、数据获取、渲染模式、后端服务等等
- Next.js 不但处理 React 所需的工具和配置，还提供额外的功能和优化，比如：
  - UI构建， CSR、SSR、SSG、ISR 渲染模式，Routing、Data Fetching等等。
- 中文官网：https://www.nextjs.cn/docs/getting-started
- 英文官网：https://nextjs.org/docs/getting-started

Next.js



Next.js 发展史

- Next.js 于2016 年 10 月 25 日首次作为开源项目发布在GitHub上，最初是基于六个原则开发的：
  - 开箱即用、无处不在的JS、所有函数用JS编写、自动代码拆分和服务器渲染、可配置数据获取、预期请求和简化部署。
- Next.js 2.0 于 2017 年 3 月发布，改进后的版本让小型网站的工作变得更加容易，还提高了构建和热模块替换效率。
- 7.0 版于 2018 年 9 月发布，改进了错误处理并支持 React 的上下文 API。升级到了webpack4
- 8.0 版于 2019 年 2 月发布，第一个提供 Serverless 部署的版本。
- 2020 年 3 月发布的 9.3 版包括各种优化和全局Sass和 CSS 模块支持。
- 2020 年 7 月 27 日，Next.js 9.5 版发布，增加了增量静态再生成、重写和重定向支持等新功能。
- 2021 年 6 月 15 日，Next.js 版本 11 发布，其中包括：Webpack 5 支持
- 2021 年 10 月 26 日，Next.js 12 发布，添加了 Rust 编译器，使编译速度更快
- 2022 年 10 月 26 日，Vercel 发布了 Next.js 13。
  - 带来了一种新的路由模式，增加了app目录 、布局布局、服务器组件和一组新的数据获取方法等（目前是 beta 版本）
  - 编译和压缩等由 Babel + Terser 换为 SWC（ Speedy Web Compiler ），构建工具增加了 Turbopack。

Next.js 特点

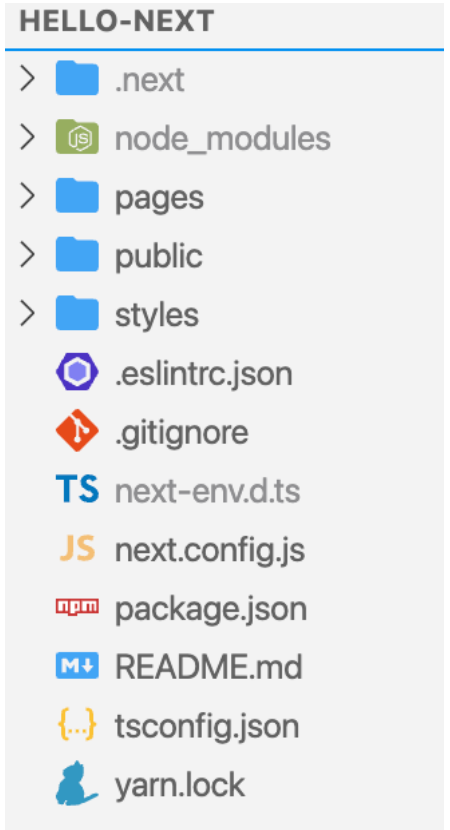
- 开箱即用，快速创建：
  - Next.js 已经帮我集成好了各种技术栈，比如：React、webpack、路由、数据获取、SCSS、TypeScript等等
  - 也提供了专门的脚手架：create-next-app
- 约定式路由（目录结构即路由）
  - Next.js和Nuxt3一样，所有的路由都是根据 pages目录结构自动生成。但在 Next.js 13 beta 版本增加了app目录。
- 内置CSS模块和Sass支持：
  - 自从Next.js 9.3 以后就内置了CSS模块和Sass支持，也是开箱即用
- 全栈开发能力：
  - Next.js不但支持前端开发，还支持编写后端代码，比如：可开发登录验证、存储数据、获取数据等接口
- 多种渲染模式：支持CSR、SSR、SSG、ISR等渲染模式，当然也支持混合搭配使用
- 利于搜索引擎优化：
  - Next.js支持使用服务器端渲染，同时它也是一个很棒的静态站点生成器，非常利于SEO和首屏渲染

Next.js VS Nuxt3

- Next.js 和 Nuxt3的相同点
  - 利于搜索引擎优化，都能提高首屏渲染速度
  - 零配置，开箱即用
  - 都支持目录结构即路由、支持数据获取、支持TypeScript
  - 服务器端渲染、静态网站生成、客户端渲染等
  - 都需要 Node.js 服务器，支持全栈开发
- Next.js 和 Nuxt3区别：
  - Next.js 使用的是React技术栈：React 、webpack 、express 、node.....
  - Nuxt3 使用的是Vue技术栈：Vue、webpack、vite、h3、nitro、node.....
  - Nuxt3 支持组件、组合 API、Vue API等自动导入，Next.js则不支持
  - Next.js 社区生态、资源和文档都会比Nuxt3友好（star数： Nuxt3 -> 41.6k 和 Next.js -> 96.8k ）
- Next.js 和 Nuxt3如何选择？ 个人建议如下：
  - 首先根据自己擅长的技术栈来选择，擅长Vue选择Nuxt3，擅长React选择Next.js
  - 需要更灵活的，选择Next.js
  - 需要简单易用、快速上手的，选择Nuxt3

Next.js 13 环境搭建

- 在开始之前，请确保您已安装推荐的设置：
  - Node.js （要求 Node.js 14.6.0 或 更高版本。 ）
  - Git（window下可以用其随附的 Git Bash 终端命令）
  - Visual Studio Code
- 创建一个项目，项目名不支持大写
  - 方式一： npx create-next-app@latest --typescript
  - 方式二： yarn create next-app --typescript
  - 方式三： pnpm create next-app –typescript
  - 方式四：npm i create-next-app –g && create-next-app
- 运行项目
  - npm run dev
  - yarn dev
  - pnpm dev



Next.js 目录结构

```
├── hello-next  # Next.js项目名称
    ├── pages  # 定义页面文件夹，路由会根据该页面目录结构和文件名自动生成
    |   ├── _app.tsx  # App组件，应用程序的入口组件
    |   ├── api  # 编写后台接口的文件夹
    |   |   └── hello.ts # 定义了一个接口，接口地址为：/api/hello
    |   └── index.tsx # 项目的首页
    ├── public # 静态资源目录，不参与打包
    |   ├── favicon.ico
    |   └── vercel.svg
    ├── styles  # 编写样式目录
    |   ├── Home.module.css  # 局部css module样式
    |   └── globals.css # 全局样式，需要在_app.tsx中导入
    ├── next-env.d.ts # Next.js 专有的类型声明文件。不应该删除它或编辑它，也不需要提交到git仓库中
    ├── next.config.js # 可定制 Next.js 框架的配置，比如：环境变量、重定向、webpack等
    ├── package-lock.json # 项目依赖库版本的锁定
    ├── package.json # 项目的描述文件
    ├── README.md  # 项目简介
    └── tsconfig.json # TypeScript的配置文件
```

入口App组件（_app.tsx）

- _app.tsx是项目的入口组件，主要作用：
  - 可以扩展自定义的布局（Layout）
  - 引入全局的样式文件
  - 引入Redux状态管理
  - 引入主题组件等等
  - 全局监听客户端路由的切换

```tsx
import "../styles/globals.css";
import type { AppProps } from "next/app";

export default function App({ Component, pageProps }: AppProps) {
  return <Component {...pageProps} />;
}
```

```tsx
useEffect(() => {
  const handleRouteChange = (url: string) => {
    console.log(`App is changing to ${url}`);
  };
  router.events.on("routeChangeStart", handleRouteChange);
  return () => {
    router.events.off("routeChangeStart", handleRouteChange);
  };
}, []);
```

ts.config.json 的配置

- Next.js默认是没有配置路径别名的，我们可以在ts.config.json中配置模块导入的别名：
  - baseUrl ：配置允许直接从项目的根目录导入，比如： import Button from 'components/button'
  - paths：允许配置模块别，比如： import Button from '@/components/button'

```
"compilerOptions": {
  "baseUrl": ".",
  "paths": {
    "@/assets/*": ["assets/*"],
    "@/components/*": ["components/*"],
    "@/styles/*": ["styles/*"],
    "@/pages/*": ["pages/*"]
  }
}
```

  - 注意：如生效可以重启编辑器

环境变量（.env*）

```
# .env
HOSTNAME=localhost
PORT=8080
HOST=http://$HOSTNAME:$PORT
```

- 定义环境变量的4种方式：
  - .env：所有环境下生效的默认设置
  - .env.development：执行 next dev 时加载并生效
  - .env.production ： 执行 next start 时加载并生效
  - .env.local ：始终覆盖上面文件定义的默认值。所有环境生效，通常只需一个 .env.local 文件就够了（常用存储敏感信息）
- 环境变量定义语法（支持变量，例如 $PORT ）：
  - 大写单词，多个单词使用下划线，比如：DB_HOST=localhost
  - 添加 **NEXT_PUBLIC_ 前缀**会额外暴露给浏览器，比如：NEXT_PUBLIC_ANALYTICS_ID=aaabbbccc
- 环境变量的获取：
  - .env 文件中定义环境变量会加载到 process.env 中。两端都可直接通过 process.env.xxx 访问使用（不支持解构）
- 注意事项：
  - 由于 .env、.env.development 和 .env.production 文件定义了默认设置，需提交到源码仓库中。
  - 而 **.env.*.local 应当添加到 .gitignore** 中，因为这类文件是需要被忽略的。

Next.js配置（next.config）

- next.config.ts 配置文件位于项目根目录，可对Next.js进行自定义配置，比如，可以进行如下配置：
  - reactStrictMode：是否启用严格模式，辅助开发，避免常见错误，例如：可以检查过期API来逐步升级
  - env：配置环境变量，配置完需要重启
    - ✓ 会添加到 process.env.xx 中
    - ✓ 配置的优先级： next.config.js中的env > .env.local > .env
  - basePath：要在域名的子路径下部署 Next.js 应用程序，您可以使用basePath配置选项。
    - ✓ basePath：允许为应用程序设置URl路径前缀。
    - ✓ 例如 basePath=/music，即用 /music 访问首页，而不是默认 /
  - images：可以配置图片URL的白名单等信息
  - swcMinify：用 Speedy Web Compiler 编译和压缩技术，而不是 Babel + Terser 技术
- 更多的配置： https://nextjs.org/docs/api-reference/next.config.js/introduction