Sofascore Frontend Academy Lecture 07,
May, 2023

Zeplin. Next.js

Alen Murtić

Sofascore

TABLE OF CONTENTS

01

Zeplin

Zeplin

What is Zeplin?

- The software with which designers in Sofascore deliver design specifications
- Reusable design elements like components, colors, and text styles
    - Based on components - loosely similar to how we organize components
- Provides informative CSS and variables used
    - Why only informative? Same issues can be solved in multiple ways

Zeplin in Sofascore (and your final project)

- Designer delivery divided into mockups and specifications (spec board)
- Mockups - how components are organized into pages/screens/tabs/components
  - Authority on copy (i.e. text labels)
  - Not a full authority on domain issues (e.g. states for match cells, etc.) - it’s **mock** data
- Specifications - describe basic building blocks
  - Color variables and typography overview, assets
  - Common components such as Button, panels
  - Overview of states of basic components (e.g. states for match cells)
    - More authoritative than mockups, still not everything is covered

Zeplin usage example

Sofascore

02

Next.js

Next.js

Motivation

- React by default fetches all JS to the client and then starts rendering
  - User gets empty page (or app shell)
  - Slower time to interaction because all user can initially see are placeholders
  - Users with slower devices and/or internet speed can have double the amount of slowness
- Server-side rendering content, especially metadata, is optimal for SEO
- Entered: Next.js

Next.js

Next.js background

- Started as a server-side rendering library for React
  - Implemented core SSR features and allowed extending them via custom Express.js server logic (e.g. translations logic)
- Became full-fledged React SSR framework
  - Added static-site generation (before that, Gatsby was the best SSG framework)
  - Removing customization ability for own solutions

Next.js

Next 13 and React 18

- Next 13 introduces many new features and changes
  - Blogpost and more docs
  - In alignment with server components introduced in React 18
  - Currently between beta and production level
- Given how Sofascore uses Next.js, those changes won't give us new features, but could introduce potential issues
  - We still use Next 12
- Because of that, at this year's academy, we will be using Next 12

Next.js

Next 13 and React 18

- Next 13 introduces many new features and changes
  - Blogpost and more docs
  - In alignment with server components introduced in React 18
  - Currently between beta and production level
- Given how Sofascore uses Next.js, those changes won't give us new features, but could introduce potential issues
  - We still use Next 12
- Because of that, at this year's academy, we won't be using new beta features

Next.js

Getting started

- Instead of using create react-app, use the following command:
  - yarn create next-app [app name] --typescript - default template
  - Alternative: template with styled components
  - Neither is perfect for our use-case, but this lesson will use official template
- Project structure is different than classic React
  - All generated files are in pages directory or inside src/pages
- Server-side and client-side routing and rendering out of the box
- Important: **disable strict mode in next.config.js file**
  - Among other things, it runs useEffect twice on render

Next.js

Pages

- Two types of pre-rendering:
- 1. Static generation - HTML is generated at build time
  - Definite (final) number of pages
- 2. Server-side rendering - HTML is generated on each request
  - Indefinite (unknown) number of pages based on parameters
- Pages folder offers routing out of the box
- _app.tsx file - all components are wrapped in its default export
  - The place to add SWRConfig or similar things
- index.tsx file - default page

Next.js

Static pages

- Without data - generates single HTML file
  - Example: pages/about.tsx
- With data - HTML + JSON props - two options:
  - Fixed path, content depends on external data - getStaticProps
    - Example - pages/privacy.tsx
  - Variable path (dynamic routing) - getStaticPaths + getStaticProps
    - Example - pages/country/[alpha2].tsx
    - The idea is to have a limited number of pages, e.g. SofaScore motorsport categories
- Still can use client-side rendering

03

Server-side rendering with next.js

SSR Basics

- HTML is not created at build time, but at each request
- Must implement method getServerSideProps
  - Fetch from dynamic routes
- Default export is your page component
- Client vs server
  - useEffect is run on client
  - useSWR is run on client
  - console.log(...) on server is written in the terminal running the app
  - Most of other code is run on server
  - const isServer = typeof window === 'undefined'

Client vs server

- useEffect is run on client
- useSWR is run on client
- console.log(...) on server is written in the terminal running the app
  - Important for debugging
- Most of other code is run on server
- const isServer = typeof window === 'undefined'
  - Sometimes this is not enough and we need custom useIsServer hook

Routing

- Next gives you routing by default
  - Link components instead of a
  - useRouter() hook
    - const router = useRouter() and router.push(path)
  - Translated routes are also an option
- More on routing

Next.js SSR example - ssr.md

Sofascore

Thank you for your attention!

Sofascore