BTI425/WEB422 - Web Programming for Apps and Services

Lecture Recap:

Week 12 – Performance Optimizations

Agenda

- Analyzing Performance
- Improving / Optimizing Performance

Analyzing Performance

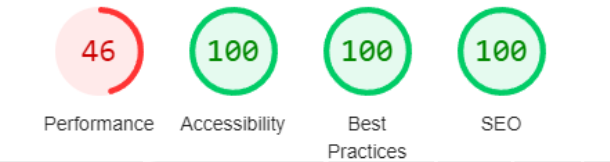
- Core Web Vitals
  - Performance metrics, similar to KPIs for business
  - A subset of Web Vitals, and currently consists of three metrics that measure:
    - loading, interactivity, and visual stability.
  - These metrics are:
    - Largest Contentful Paint (LCP), First Input Delay (FID), and Cumulative Layout Shift (CLS).

Introduction to Lighthouse

- A tool from Google to measure "Core Web Vitals"
- An open-source, automated tool for improving the quality of web pages.
  - You can run it against any web page, public or requiring authentication.
  - It has audits for performance, accessibility, progressive web apps, SEO (**search engine optimization - Making your content search-friendly matters**), ...
- Lighthouse is integrated directly into the Chrome DevTools
  - available in the "Lighthouse" panel - F12
  - you may wish to access lighthouse in one of the other methods
    - Using the Node CLI
    - As a Node Module
    - Using the online tool: PageSpeed Insights
  - NOTE: it is also available as a GitHub action - use it in our CI pipeline.

Analyzing Page Load

- To begin, download the Example Code
- The app (Film Collection) pulls the films data from an "/api/movies" endpoint (specified in our "pages/api/movies.js" file)
  - Check the "Performance", "Accessability", "Best Practices" and "SEO" for Categories.
  - Click the Analyze Page Load button and wait for the audit to finish.



- Testing a production build, run:
  - npm run build
  - npm run start

Improving / Optimizing Performance

Optimization: Using Next <Image> to replace html <img>:

- Improved Performance, Visual Stability, Faster Page Loads, Asset Flexibility

- Attribute "priority":
  - cause the Image to preload
- Attributes 'width={800} height={232}'
  - original width/height in pixels
- Attribure 'sizes="100vw"':
  - which image will be downloaded
  - scrset from browser console:

<link rel="preload" as="image" image

- Remote Images
  - Set domains of the images in next.config.js

Improving / Optimizing Performance

Optimization: Dynamically Importing Libraries:

- Next.js supports "Lazy Loading" for external libraries with "import" as well as images and pages.
  - to reduce the initial bundle size and improve your performance

- Dynamically import library:
  - Remove "import" statement(s)
  - Syntax: e.g.

```
async function …
    const _ = (await import('lodash')).default;
    …
```

  - Importing/loading the library once it is required.

Improving / Optimizing Performance

Optimization: Dynamically Importing Components:

- Components can also be dynamically imported to reduce the initial bundle size and improve your performance.
- Dynamically load the "StarRating" component
  - Remove the initial import
  - Add 2 imports

```
import dynamic from 'next/dynamic';
import { Suspense } from 'react';
```

  - Dynamically import the component using the "dynamic" function
  - Add a flag in the "state"
  - Set an "onSelect" event on the flag changed to true
  - Add an eventhandler function to change flag value to true
  - Wrap the dynamically importing component using <Suspense …> which is set to be rendered when the flag is true

Improving / Optimizing Performance

Optimization: Refactoring from SSR to pre-render:

- To help reduce the time to first render and improve application performance,
  - we should try to **pre-render** as much of the page as possible (using SSR)
  - e.g.: the static list on the "Film Collection" Home page isn't likely to change frequently

- Example: Refactoring the "Film Collection" app for pre-rendering the static list on the Home page:
  - …

The End