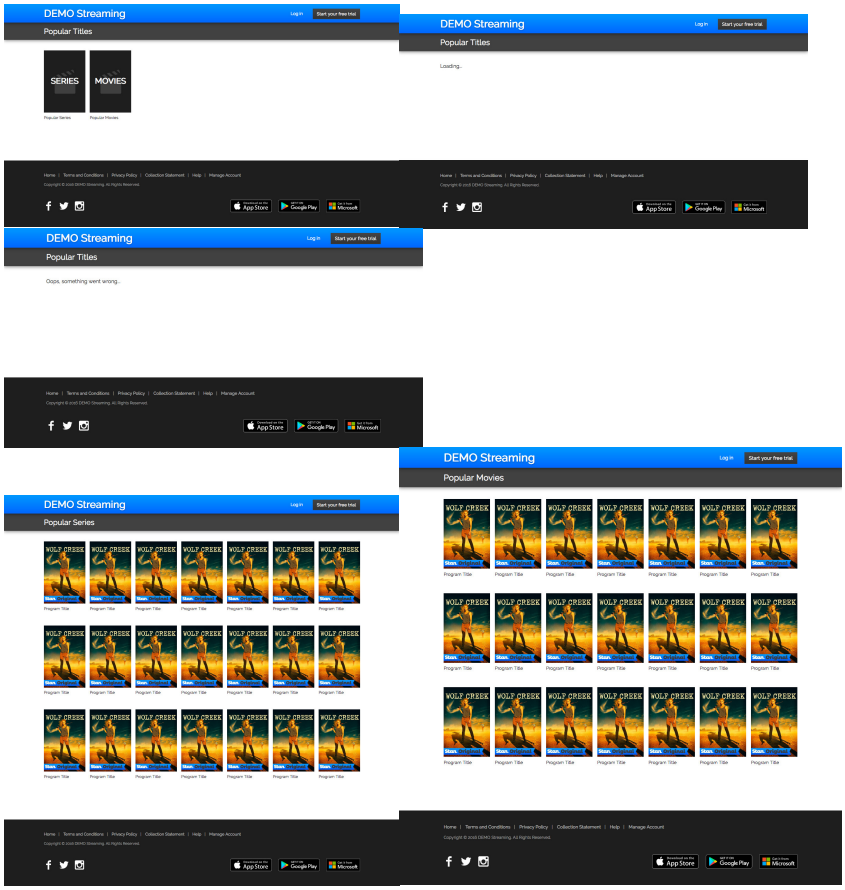
Coding Challenge

1. Introduction
2. Requirements/Assumptions
3. Technical and architectural choices
4. Imported Libraries
5. Testing
6. Improvements
7. What would you do differently if you were allocated more time?
8. Appendix

Introduction

Implement in react the screens show below in react using a minimum of external dependencies and css libraries.



Requirements / Assumptions

A number of assumptions were made in developing the coding challenge. They are listed below:

- The supplied poster art images are oversized. No attempt was made to optimize or reduce the size.
- The app was created using create react app. There was no need to eject at this time.

- No designs are supplied for mobile and tablet. A best guess was made at building the responsive layouts for those devices.
- Targeted modern browser.
- Title sort removes “The” and “A” from start of title.
- “We prefer it if you did not use any third party CSS frameworks”.
- “We also prefer the use of minimal dependencies”.
- Bundle size/tree shaking has not been optimized.
- Build folder is build/ instead of dist/. To accomplish this one would eject the create react app and change the output folder in webpack config. Does not seem necessary to accomplish this at this time.

Technical and Architectural choices

Components

The page is broken down into major visual and logical components. Several elements persist between routes. GlobalHeader GlobalFooter. The varying page content is contained in a react router switch statement. The content in this switch statement varies base on route. In this simple design there is a component Page. It can accept either a list of items or a data source and selector as its props. Styled components was used for the styling of the components the components exist under the directory “components”. The reason for this component structure is to maximise component reuse and testability. A directory called modules was created. This is to differentiate reusable components from components that might be used in a one off capacity. The rule would be that modules can import components and modules but components can only import from components.

Data fetching / State management

Fetch suspense was used for the fetching of the feed data. There was no attempt at integrating the fetched data into the redux store or persisting data. In a real application this would be implemented as needed. Data fetching using this method is a bit unorthodox at the moment but I see the future of react using this design pattern more frequently. I believe relay and Apollo are already using it. Another way would be to fetch the data in useEffect or componentDidLoad.

A logged in state is provided to demonstrate how redux would be used in the application. It is accessed by clicking log in. The only change is that when you are logged in the header will now show a “log out” button.

Code splitting

The main application is lazy loaded while a logo “Stream Co” is displayed. This does not work as well as expected. It takes a while for the logo to appear and only displays shortly before the main application is loaded. This should be developed further to work as expected. As more modules are added they can be lazy loaded as required.

Styling

Two styling approaches was considered. Using a css processor such as Sass and styled components. Styled components was chosen due to it being javascript native. Variables and code can be shared as needed between application and styling. Some responsiveness was coded for mobile/tablet.

Image loading

A placeholder image is shown while the image is loading. Only images in view will be loaded. Once the image is loaded a class is added so the image appears to fade in giving a pleasing

visual effect. A hover state is implemented to give more feedback to the user. The images were not resized and optimized which they would be in a production application. It doesn't seem to be the point of the coding challenge to do this

Libraries

Fetch-suspense

Using the Suspense component requires the use of a fetch library that implements the suspense api.

React router

React router is a widely used routing solution.

React redux

React redux was imported as a tried and true state management solution

Reselect

Reselect is used to return computed values from the store. The selectors are memoized preventing unnecessary calculation of outputs.

Styled components

Styled components was used for styling

Cypress

Cypress is used for integration testing

Testing

Some basic unit tests are created with jest for the reducer and actions. Unit tests are set up for the page filters/selectors.
Yarn test
Some basic integration tests are set up using cypress. For ui integration tests tend to be more useful.
Yarn cypress:open

Improvements

Some improvements:

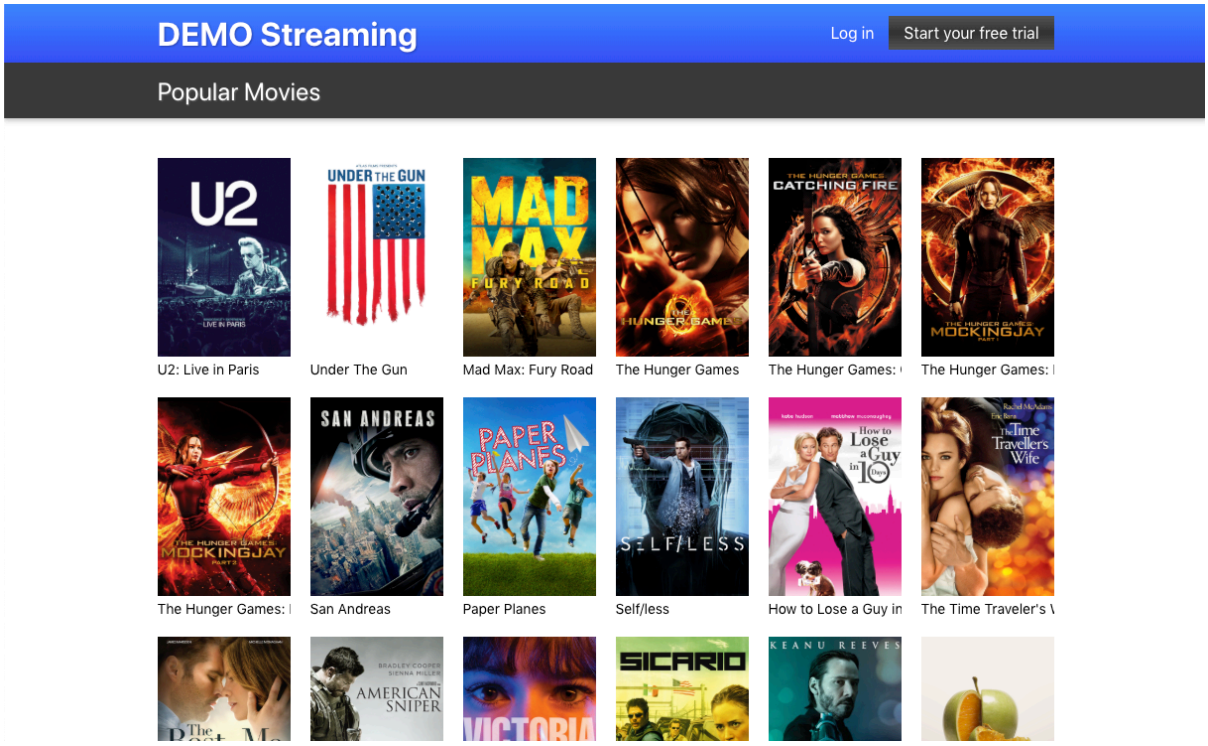
- Parameterize styling constants such as color and sizes. These could be in a file exporting the values.
- Serve optimized images.
- Improve the mobile and tablet responsive design
- Improve the state management/data persistence
- Implement PWA
- Selector could have more parameters

What would you do differently if you were allocated more time?

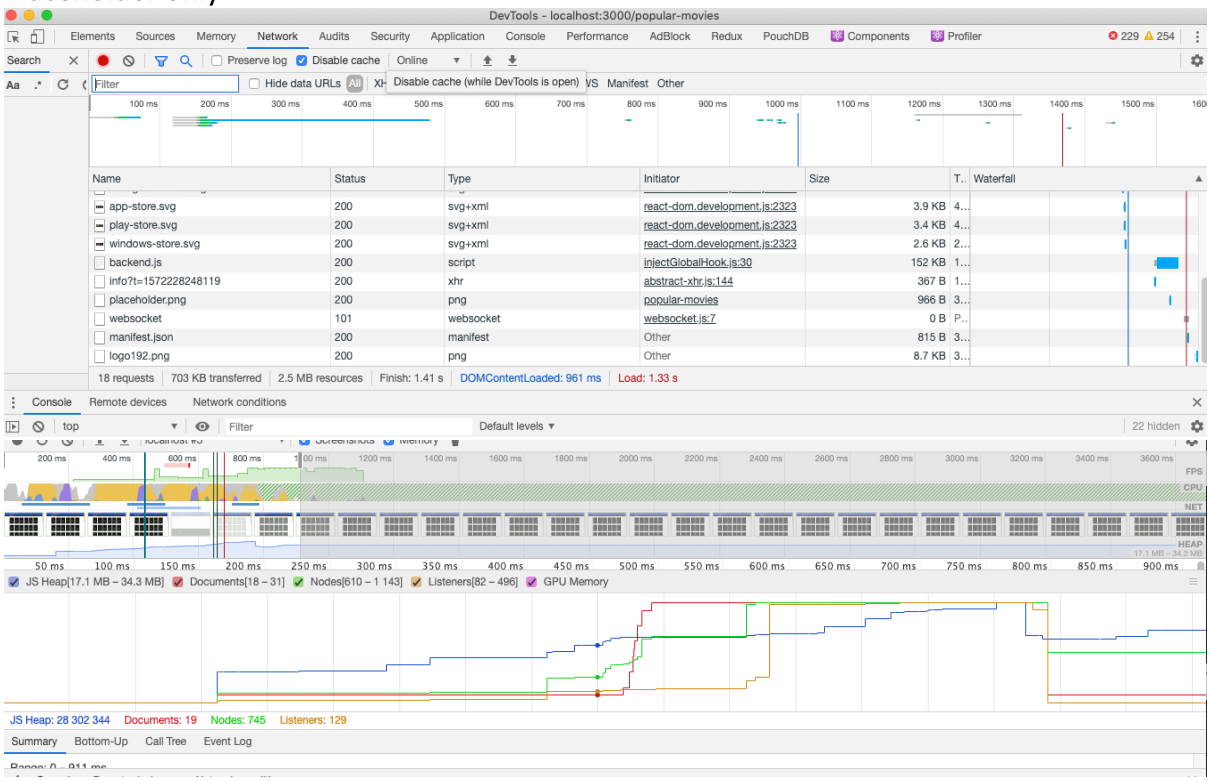
- Styling improvements for mobile and tablet.
- Error handling.
- Set up better integration with api feed and persistence.
- Set up more better animations and transitions.

Appendix

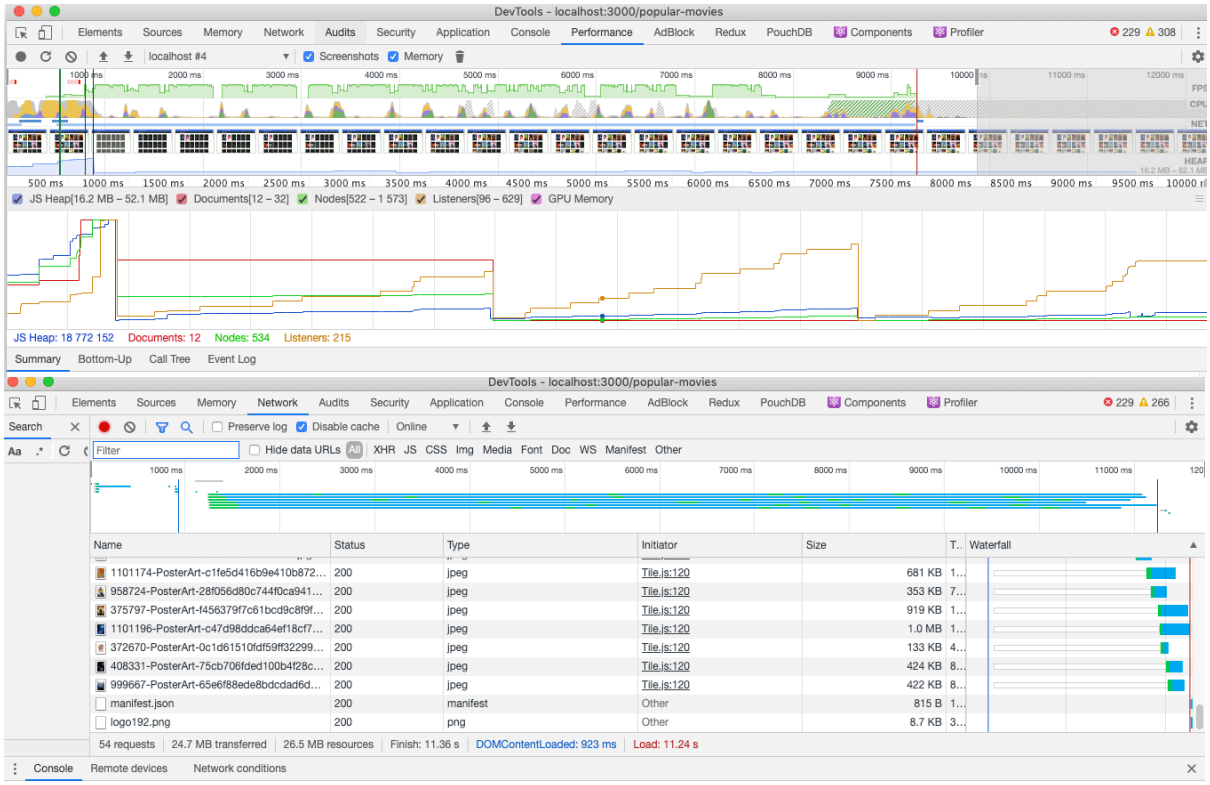
Image loading profiling



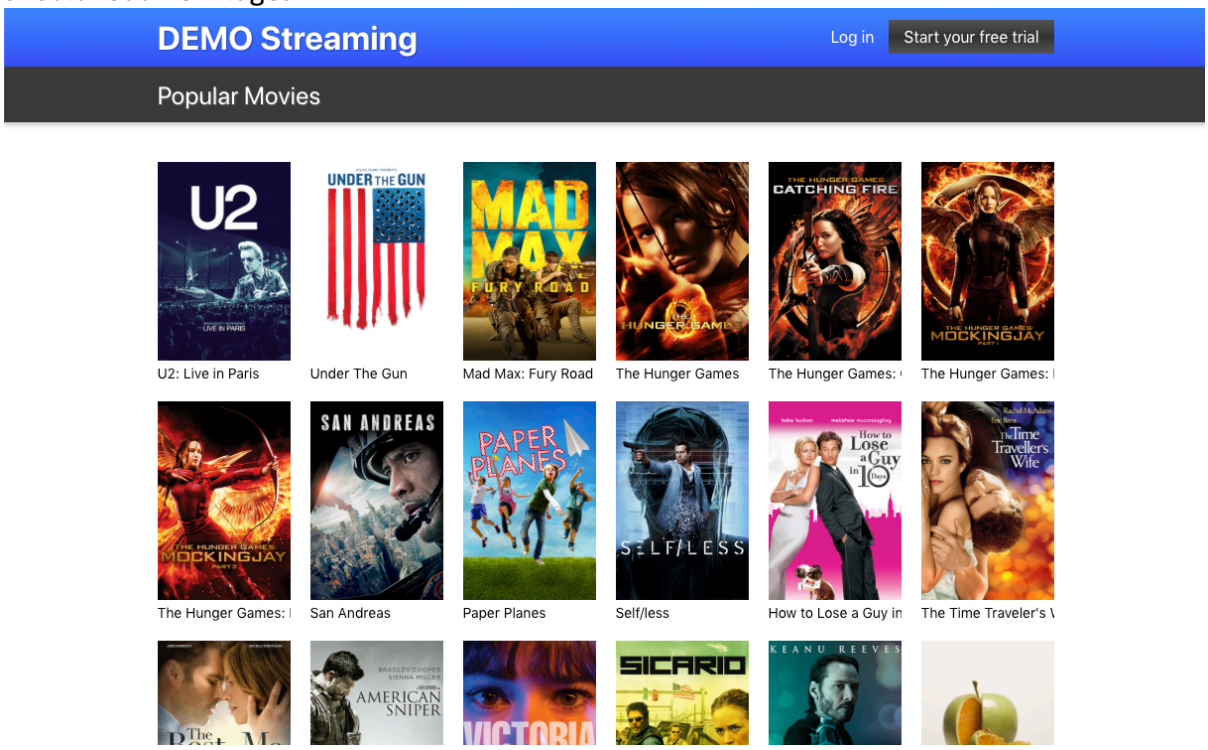
Placeholder only



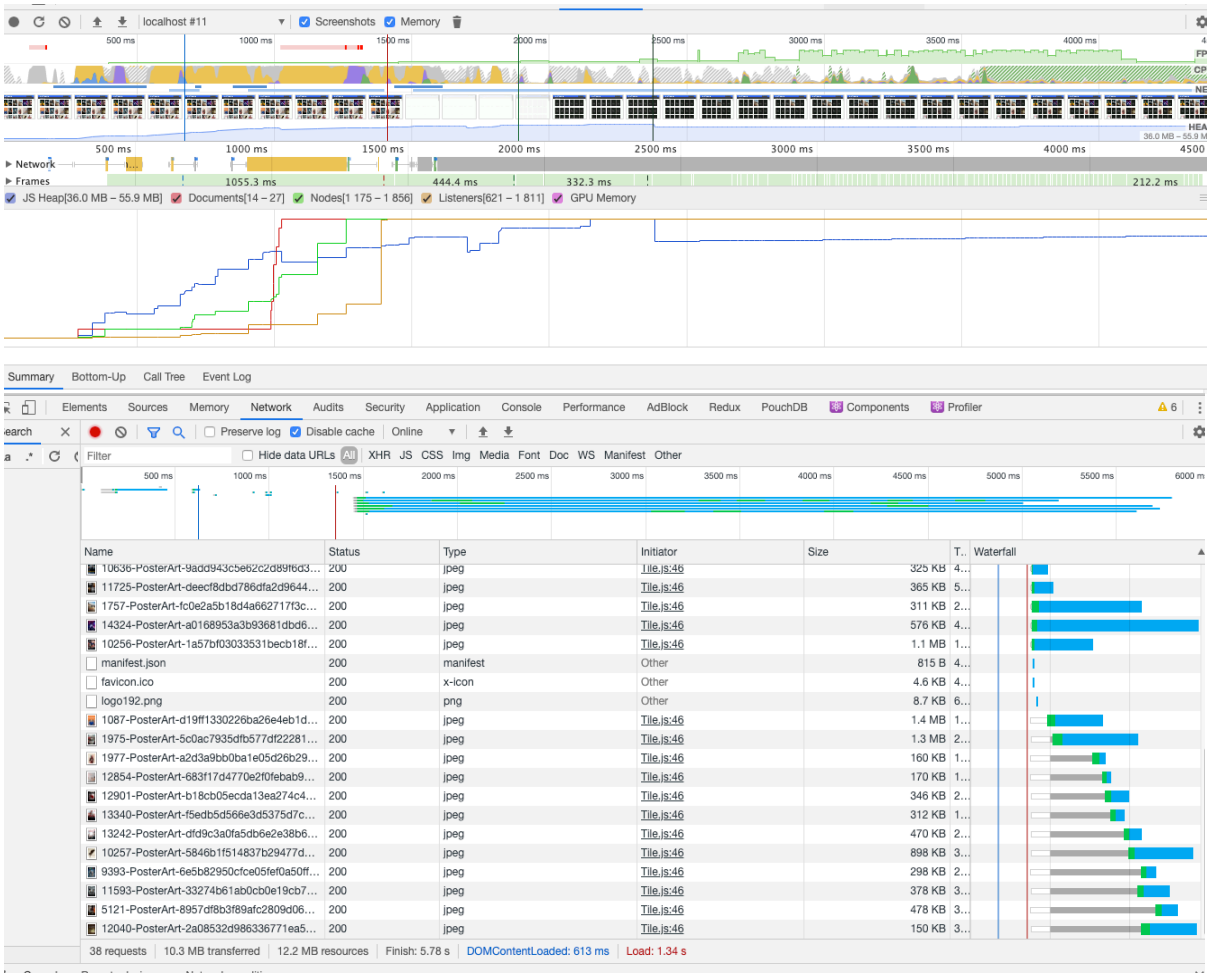
all images



lazyloaded
should load 18 images



lazyloading implemented


localhost #11
Screenshots
Memory
Network
Frames
1055.3 ms
444.4 ms
332.3 ms
212.2 ms
JS Heap[36.0 MB – 55.9 MB]
Documents[14 – 27]
Nodes[1 175 – 1 856]
Listeners[621 – 1 811]
GPU Memory
Summary
Bottom-Up
Call Tree
Event Log
Elements
Sources
Memory
Network
Audits
Security
Application
Console
Performance
AdBlock
Redux
PouchDB
Components
Profiler
Preserve log
Disable cache
Online
Filter
Hide data URLs
XHR JS CSS Img Media Font Doc WS Manifest Other
Name
Status
Type
Initiator
Size
Waterfall
38 requests
10.3 MB transferred
12.2 MB resources
Finish: 5.78 s
DOMContentLoaded: 613 ms
Load: 1.34 s