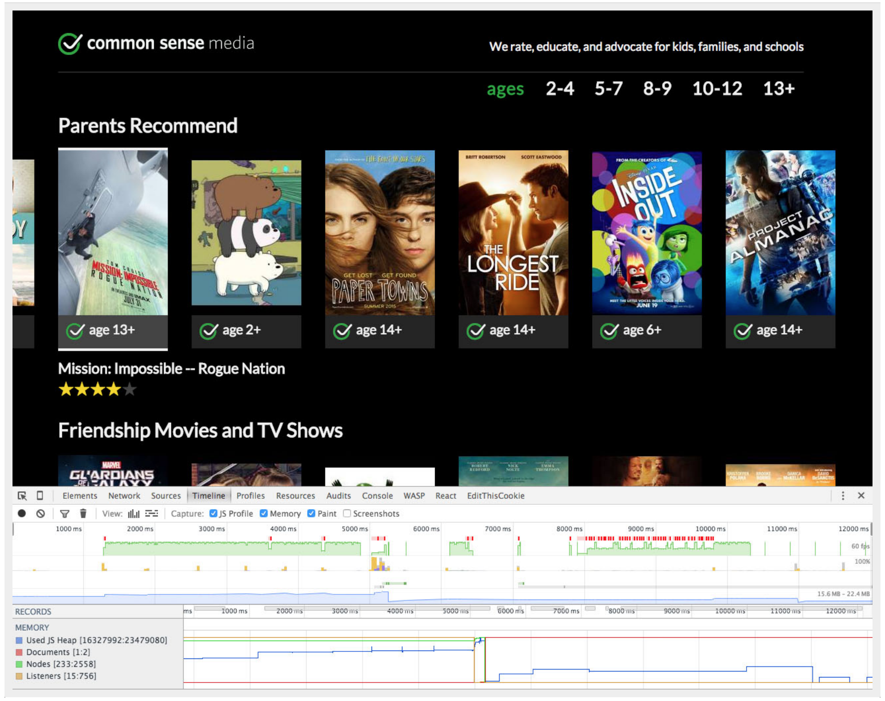
FRONT END PERFORMANCE

IN THE

COMCAST APP


common sense media
We rate, educate, and advocate for kids, families, and schools
ages 2-4 5-7 8-9 10-12 13+
Parents Recommend
age 13+
age 2+
age 14+
age 14+
age 6+
age 14+
Mission: Impossible -- Rogue Nation
Friendship Movies and TV Shows
Elements Network Sources Timeline Profiles Resources Audits Console WASP React EditThisCookie
View: Capture: JS Profile Memory Paint Screenshots
RECORDS
MEMORY
Used JS Heap [16327992:23479080]
Documents [1:2]
Nodes [233:2558]
Listeners [15:756]

FRONT END PERFORMANCE

- Review the Comcast App
- Profiling performance bottlenecks with Chrome DevTools
- Front end problems and solutions

COMCAST APP

https://comcast-client-production.commonsense.org/

- HTML5 app running in a browser on Xfinity X1 cable box
- Browser is a proprietary build of WebKit, most resembles Safari
- Remote control buttons map to javascript keyboard events

COMCAST APP

NAVIGATING THE APP

- Home page - arrow around carousel, products, legal section and filters
- Feature page - filters, product page
- Advice Videos page - watch advice video
- Home Products page - watch review video
- Legal pages

COMCAST APP

FRONT END ARCHITECTURE

- Single Page Application - All HTML, CSS and JS loaded once. Data loaded on demand via AJAX requests.
- React - performant UI rendering
- Browserify - compiles JS dependencies into one final JS file
- Gulp - minifies code, compiles JSX to javascript, updates assets
- Images - autoresized nightly from CSM Drupal production source images

COMCAST APP

PROJECT HISTORY

Angular + jQuery prototype - Rich UI with:

- Animated opacity fading
- Box shadows around products
- Zooming image scaling.

"Worked fine on my MacBook Pro."

COMCAST APP

PROJECT HISTORY

Angular + jQuery prototype

- Rendered one frame then app would be unresponsive
- Comcast webkit browser CPU was profiled to be roughly 20 times less powerful, with a memory capacity 20 times smaller than my setup with a MacBook Pro running Chrome
- Hardware was roughly equivalent to a netbook from 2010
- ...but with a powerful modern GPU

COMCAST APP

PROJECT HISTORY

Mega Refactors

- CSS Refactor - got rid of box shadows, image zooming and opacity fading
- Tried to make Angular work.
- Tried to make Angular work.
- Tried to make Angular work.
- Project put on hold.
- Discovered React.

COMCAST APP

PROJECT HISTORY

React prototype to final app

- Angular + React prototype - Works. Proves DOM Manipulation, "Layout Thrashing" and Reflow is causing slowness.
- Replaced Angular services with Flux Stores
- Replaced Angular $http with superagent
- Removed Angular from the stack
- Built app from prototype to final form in React

PROFILING FRONT END PERFORMANCE

CHROME DEVTOOLS

- Network - Analyze assets as they are downloaded.
- Timeline
  - Analyze script execution time (yellow)
  - Analyze reflow time (purple)
  - Analyze paint time (green)
  - Analyze memory use over time
  - View total memory used - Used JS Heap
- Rendering - watch the browser paint in realtime with visible feedback, FPS meter and scrolling performance analysis

PROFILING FRONT END PERFORMANCE

CHROME DEVTOOLS

Optimizing Reflow

- Specify Image Width & Height - Decoding and resizing images causes the browser an additional operation during reflow.
- Do not render areas that are offscreen. Use React's shouldComponentUpdate
- Keep DOM structure as stable as possible

PROFILING FRONT END PERFORMANCE

CHROME DEVTOOLS

Optimizing Paint - Avoid Costly CSS

- animation
- opacity
- gradients
- box-shadow

PROFILING FRONT END PERFORMANCE

CHROME DEVTOOLS

Optimizing Paint

- GPU acceleration
  - Move layers (divs) to the GPU for rendering by applying CSS -webkit-transform: rotateZ(360deg);
  - Scroll using translate3d vs animating the top/left CSS attributes of a div
- Paint small areas of the screen per React update
- Prevent overlapping divs in scrollable areas

PROFILING FRONT END PERFORMANCE

CHROME DEVTOOLS

Optimizing Script Execution

- Optimize hot code paths such as loops that execute on every render.
- Avoid deep property lookups of objects. Flatten deeply nested objects.
- shouldComponentUpdate prevents unnecessary script from executing.

PROFILING FRONT END PERFORMANCE

CHROME DEVTOOLS

Memory Leaks

- Keep DOM stable.
- Be sure to unmount global variables such as timers.
- Do not set numerous event listeners. Delegate event listeners when possible.

PROFILING FRONT END PERFORMANCE

RESOURCES - VIDEOS

- Speed Up Your JavaScript, Nicholas Zakas
- #perfmatters: 60fps layout and rendering
- Memory Management Masterclass with Addy Osmani
- React Fundamentals egghead.io series

PROFILING FRONT END PERFORMANCE

RESOURCES - ARTICLES

- Writing Fast, Memory-Efficient JavaScript By Addy Osmani
- Jank free, 60fps performance tips
- 10 Javascript Performance Boosting Tips from Nicholas Zakas
- Chrome Performance profiling with the Timeline
- Beware JavaScript Layout Thrashing!
- React docs
- React Advanced Performance + shouldComponentUpdate

shouldComponentUpdate

FRONT END PERFORMANCE

DISCUSSION