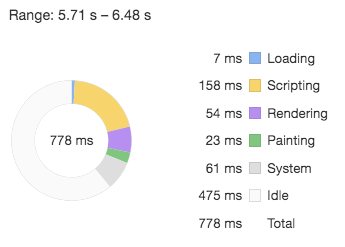
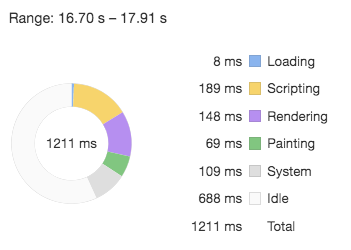
Performance Audit: Competitor Site

Having performed a Lighthouse audit on our competitor site, I have identified a number of ways in which we could optimize load time and performance. The score obtained by the site in said audit was a 47 – less than half of the possible score. By following the steps below, we could expect to raise that number substantially

Operation Analysis:

****

Tasks occupying main thread during the operation of adding a new item to the to-do list



Tasks occupying main thread during the operation of removing an item from the to-do list

Main opportunities for performance optimization on competitor site

* Minimize main thread work (12.3s total during page loading alone)
  + The large amount of time spent evaluating (7,842ms alone), parsing and compiling script can be reduced significantly, and by doing so liberate the main thread for other tasks. Numerous JS files are utilizing only a small percentage of the code included in the file (eg jquery-ui.js has 76.2% unused code) and often this unused code is comments. Action necessary: strip out all comments and reduce file size by minifying all JS, HTML and CSS files.
* Reduce time to interactive (a massive 11.2s)
  + JavaScript execution time (8.1s) may be reduced by implementing code splitting (dividing our code into several bundles which can be loaded on demand or in parallel), giving the load time a serious boost by only sending the code needed.
  + By making use of the ‘font-display’ CSS feature, we can ensure that text remains user-visible while webfonts are loaded from fonts.gstatic.com.
* Serve images in next-gen formats
  + 4,59ms can be saved by delivering /images/texture.png in an image format such as JPEG 2000, JPEG-XR or WebP due to the better compression offered by these formats.
* Improve caching policy
  + Repeat visit load times can be optimized by the use of an efficient cache policy. Static objects can have their ‘cache-control: max-age’ directive set so that they are cached for one year or more so that on subsequent requests, the browser can use the local copy rather than fetching from the server.
* Deliver critical CSS inline
  + By delivering our critical CSS (elements currently contained within style\_g.css and jquery-ui.css) inline, we eliminate render-blocking resources and shave 960ms off of our first paint.
* Reduce critical request chain depth
  + With a total of 13 chains, the critical path is too long. We must analyze the path and decide which elements are in fact critical for our page load and which elements can be deferred or marked as asynchronous. We must also minimize the download size of all crucial resources.

Pros

* Advanced, highly sophisticated functionality
* Remote data storage

Cons

* High memory consumption and slow load time
* Un-optimized code base (inefficient)
* Google Ads creates a severe drain on memory