# How Can Web Developers Optimise for Lower Power Smartphones in the Developing World?

303COM Undergraduate Presentation

Edward Prince 01/04/19

# The Problem

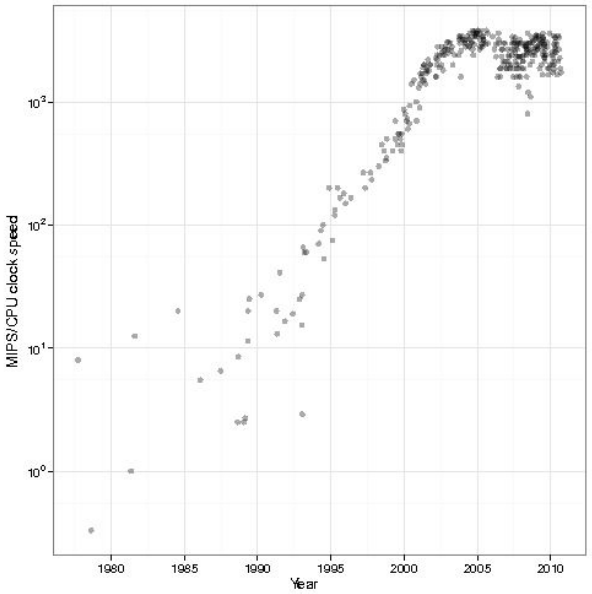
There is research being conducted into many areas of web optimisations, browsers, networks, battery etc. However, there is a distinct lack of research on optimising for low-powered CPU's rather than the current generation of CPU.

# Why should there be research conducted?

Whilst predominantly the developed world has less need for web apps that have been optimised for low-powered smartphones - it’s not the case globally. With limitations on cost of modern mobile phones - large parts of developing populations use older devices, with a lower hardware specifications than in common usage within the developing world.

# Moore’s Law

Over the last 10 years, Moore’s Law has begun to plateau - and suggests at least in the near future, the little increase in computational power may continue to preside.



# Network Speeds

In contrast to this - there has been a general shift in network speeds increase over the last 10 years in Kenya.

Should these trends continue - this would leave a fast-network/low-powered CPU future.

*Kenyan Network Speed*

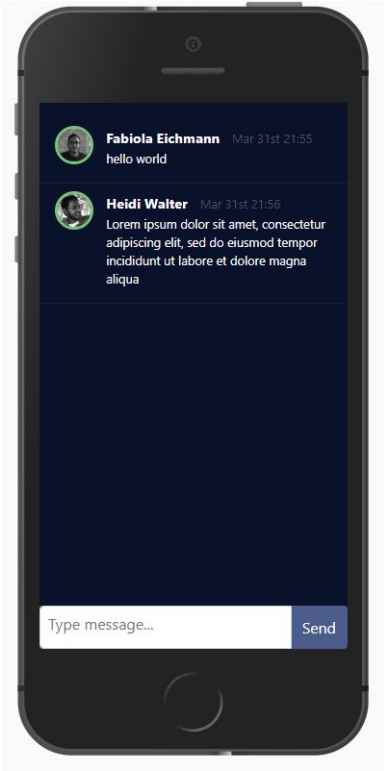
# Fast Network/Low Power CPU Future

# Carrying Out the Research

1. Create a testbed application
2. Collect data on application
3. Implement a change
4. Collect contrasting data
5. Analyse results

# Testbed Application

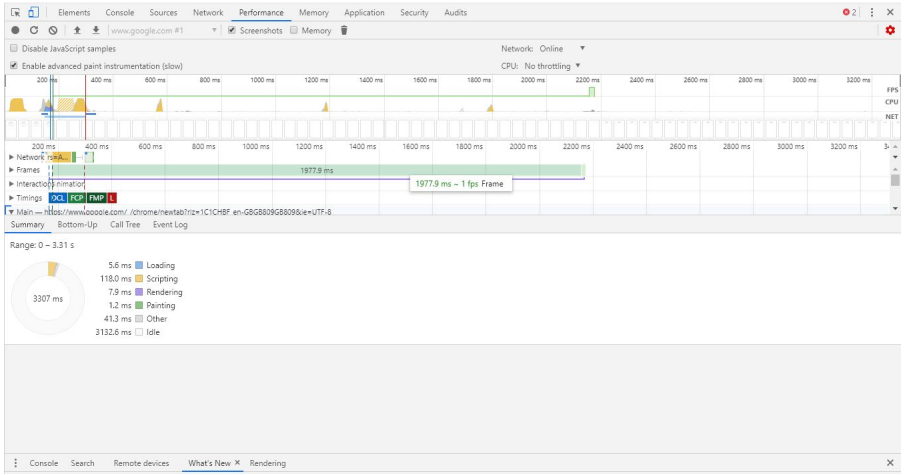
A testbed realtime chatting application - utilising modern industry-standard technologies (React, Node.js, Express.js, Socket.io) was created with which to experiment - granting the ability to gather in-depth metrics upon applying new technologies (webpack, rollup, parcel, server-decision tools).



# Google Chrome Devtools

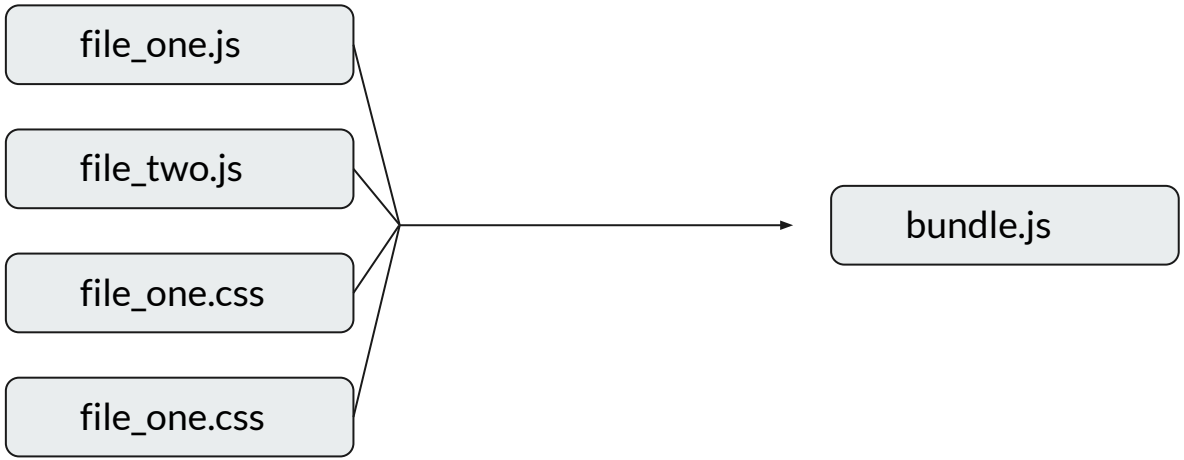
The Google Chrome Devtools is the industry standard set of tools that developers use in the browser to gain insights into how code is interacting with the browser.

There are also tools like Puppeteer to allow headless browser automation, giving easier access to more results within a given timeframe.

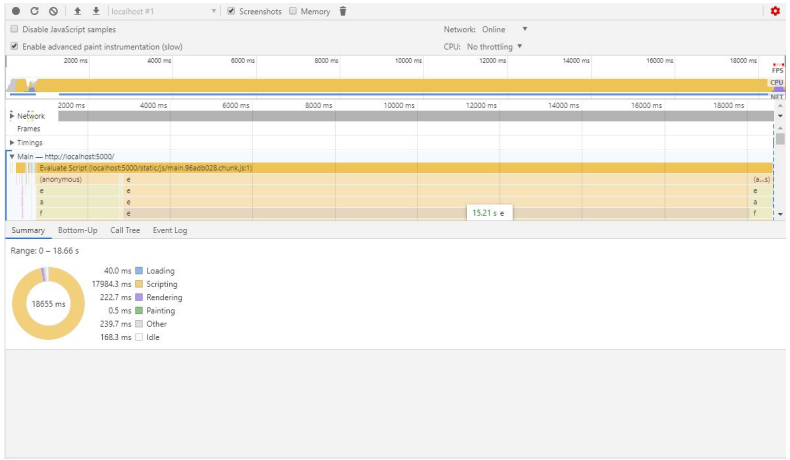


# Case 1: Bundling an Application

What is bundling?



# Data Collection with Chrome Devtools

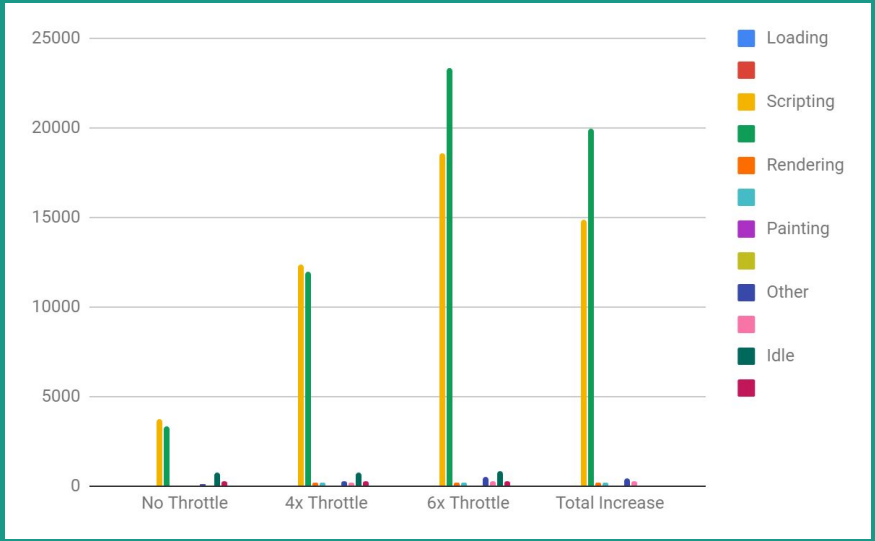



Running the app in its current state through the Devtools performance profiler yields useful data about the load times and performance metrics.

This data can then be scraped and put into a spreadsheet for further analysis.

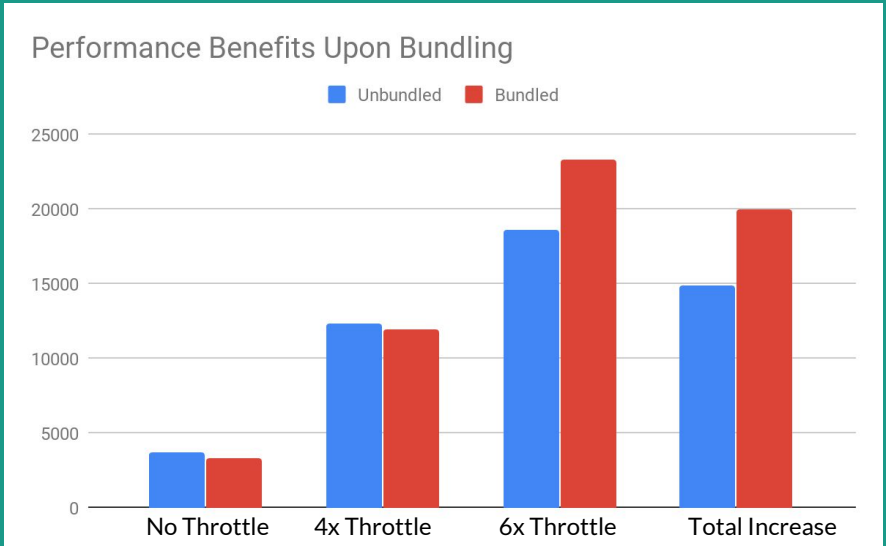
Using a custom script - much of this process could be automated for faster collection - meaning more results could be collected.

# Results




Compiling the results into a spreadsheet, and consequently into a chart showed that compared to processes that occurred in the scripting heading, loading, rendering, painting, idling, and other task performance was negligible. This gave a key area of optimisation to focus on.

# Scripting in isolation



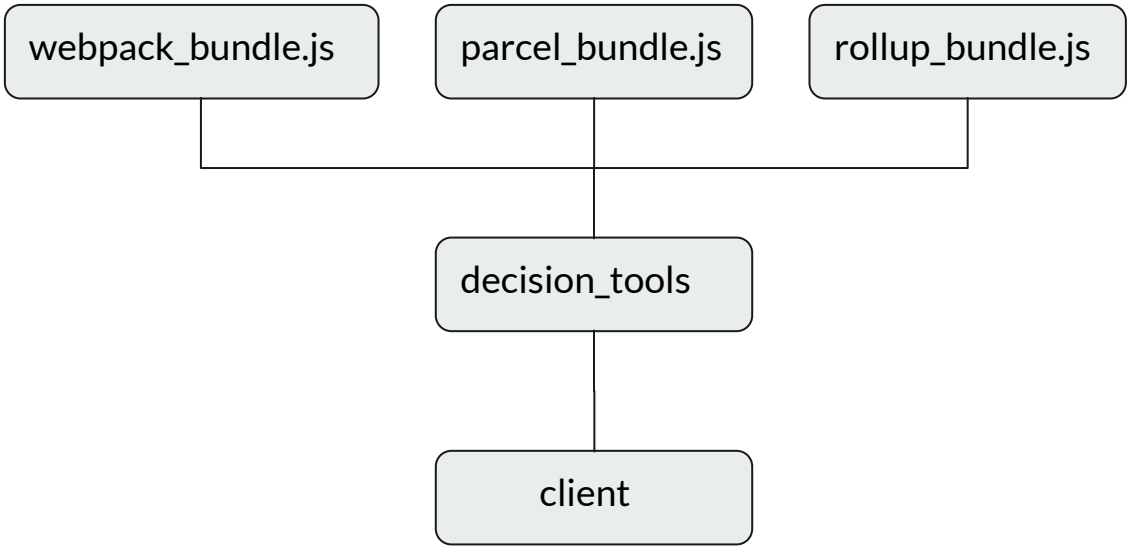

Whilst at no throttle or a 4x throttle, the bundled version does perform marginally more efficiently - at a 6x throttle, the bundled version takes a large hit, leaving the total increase from no throttle to 6x throttle a large difference.

# Still to Come

PARCEL

Blazing fast, zero configuration web application bundler

# Case 2: Server-Based Decision Tools



# Client vs Server-Based Architecture

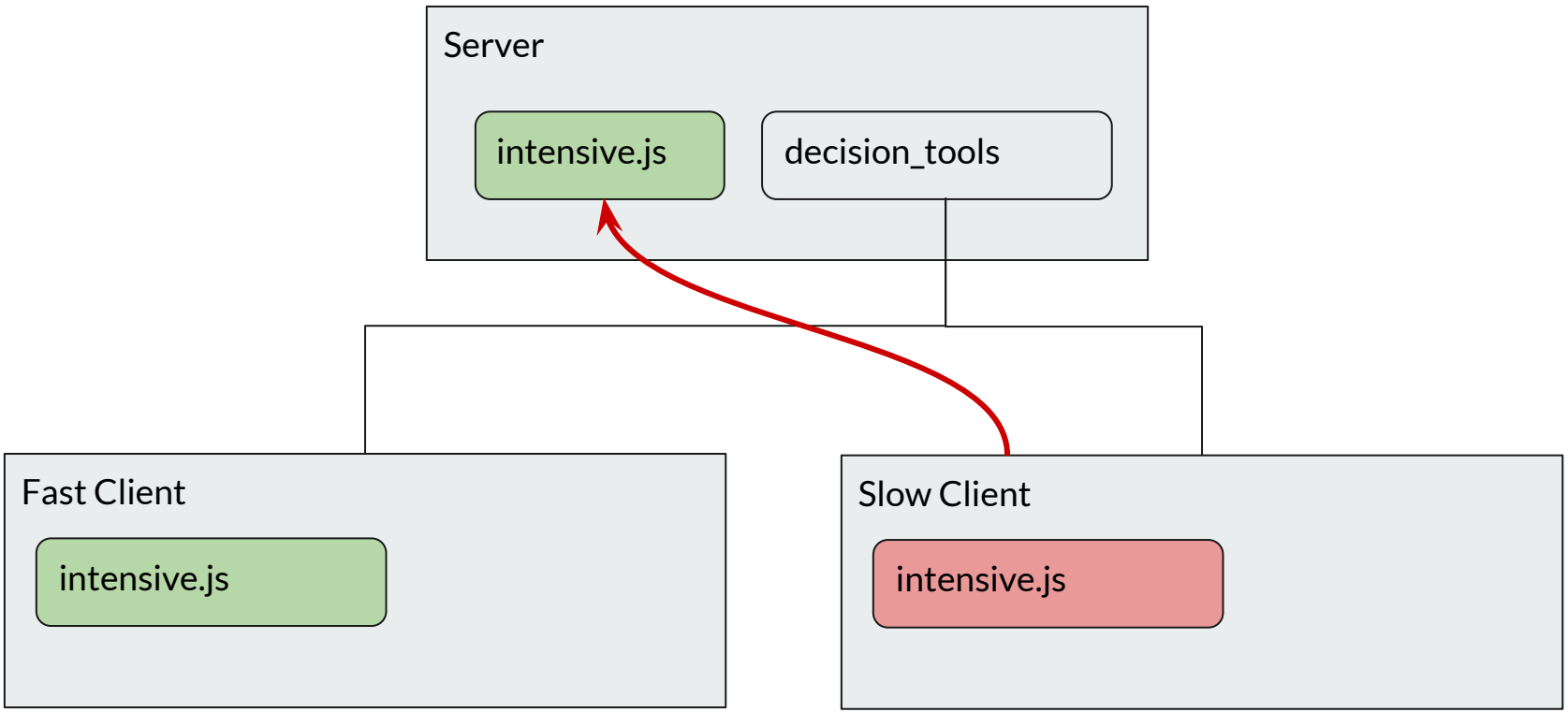
Web technology has seen shifts in desires for architecture to be client and server based. With powerful modern devices - it is becoming more common for the heavy lifting to be done on the client - leaving the server running lightly to keep high speeds with high volumes of traffic.

With low-powered devices, if developers wish for fast user experiences, then the server-based architecture become relevant again.

# Accessing the Client

JavaScript is now capable of acquiring an estimate of the processor speed that is executing the script. This means an application can tell the server basic information about the device - allowing the server to make useful decisions on what it should and shouldn't send/process.

This optimises processes for specific devices, keeping servers quick and using client-based architecture where devices are capable - and switching to server-based architecture and optimised bundles for older browsers and slower architecture.


Server
intensive.js
decision_tools
Fast Client
intensive.js
Slow Client
intensive.js

# TODO

- ~~Create testbed application~~
- ~~Decide on data collection strategy~~
- ~~Implement automated solution~~
- ~~Collect data on webpack bundling~~
- ~~Analyse results and produce charts~~
- Gather metrics for other bundlers and compare results
- Implement server-side decision tools and analyse usage results
- Complete write up and formalities (statement of originality etc.)

# Bibliography


- Gough, N. (2005) *Africa: The Impact of Mobile Phones* [online] available at <https://www.share4dev.info/telecentreskb/documents/4552.pdf> [26 January 2019]
- Iliev, I. (2014) *Front end optimization methods and their effect* [online] available from <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6859613> [03 February 2019]
- ITWeb (2018) *Africa Sees Strong Internet Growth* [online] available from <https://www.itweb.co.za/content/VgZeyvJA3V6qdjX9> [11 February 2019]
- Simonite, T. (2016) *Intel Puts the Brakes on Moore's Law* [online] available from <https://www.technologyreview.com/s/601102/intel-puts-the-brakes-on-moores-law/>
- [03 February 2019]
- Trading Economics (2017) *Kenya Internet Speed* [online] available from <https://tradingeconomics.com/kenya/internet-speed>[10 February 2019]
- Wang, H., Kong, J., Guo, Y., Chen, X. (2013) *Mobile Web Browser Optimizations in the Cloud Era: A Survey* [online] available from <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6525571> [10 February 2019]