
THE FUTURE OF
FRONT-END
PERFORMANCE
Sia Karamalegos

HI, I'M SIA


WHY DO ELEVATORS HAVE MIRRORS?

WHY SHOULD I CARE?

> *Rebuilding Pinterest pages for performance resulted in a 40% decrease in wait time, a 15% increase in SEO traffic and a 15% increase in conversion rate to signup.*

https://wpostats.com/

> *AliExpress reduced load time by 36% and saw a 10.5% increase in orders and a 27% increase in conversion for new customers.*

https://wpostats.com/

> *Speed is now used as a ranking factor for mobile searches.*

https://developers.google.com/web/updates/2018/07/search-ads-speed

MEASUREMENT AND ANALYSIS

Pareto Principle

Roughly 80% of the effects come from 20% of the causes.

Be lazy. Only optimize the worst offenders.

Which metrics matter?

- ~~Load time~~ Speed index
- Time to interactive
- Jank / responsiveness

Speed Index

Measures how quickly the page contents are visually populated

- Expressed in milliseconds
- Dependent on size of the view port
- Use webpagetest.org to measure your pages

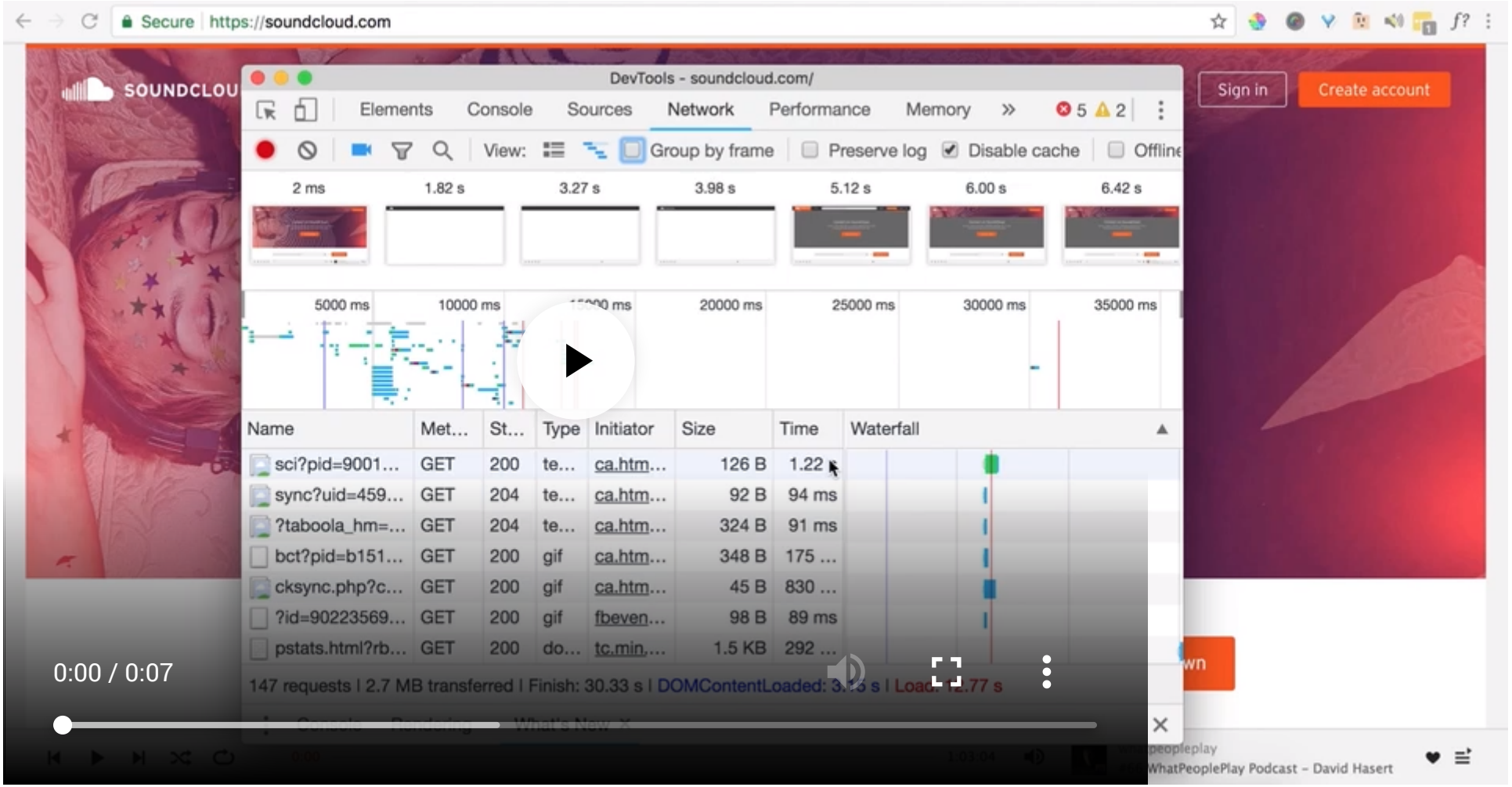
Time to Interactive

End to End Apps with Polymer by Kevin Schaaf, Polymer Summit 2017

Jank or Responsiveness

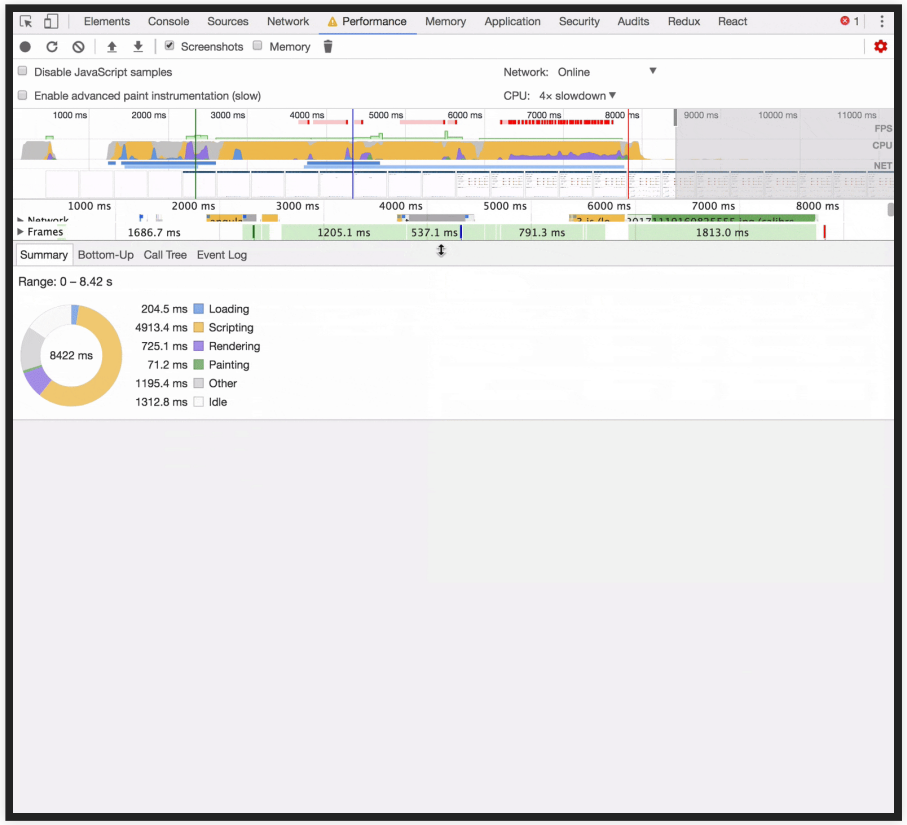
Synthetic Testing

Use WebPageTest and DevTools network tab to optimize load and speed index.



Synthetic Testing

Use DevTools performance tab to optimize responsiveness.



Real User Monitoring (RUM)

- Navigation Timing API
- Resource Timing API
- User Timing API for custom timings

https://developers.google.com/web/fundamentals/performance/navigation-and-resource-timing/
https://www.keycdn.com/blog/user-timing/

Optimize for the device and network your users have

- 2-5x difference in fastest vs slowest phones
- 75% of worldwide mobile connections on 2G or 3G
- Not just developing countries but rural areas or spotty networks like conference wifi
- Use Google Analytics data to profile your users and configure webpagetest.org to reflect them more closely
- Set performance budgets using webpack

https://infrequently.org/2017/10/can-you-afford-it-real-world-web-performance-budgets/

BIGGEST BYTES

> *Images account for 60% of the bytes on average needed to load a webpage.*
>
> Google

Image Optimization Toolbox

- Use the right image type (png vs jpg, gif vs video).
- Use the right size and src sets, and webpack loaders to auto-build src sets.
- Compress images with a tool like ImageOptim, or use a webpack plugin to auto-optimize them for you.

https://www.udacity.com/course/responsive-images--ud882
https://survivejs.com/webpack/loading/images/#optimizing-images

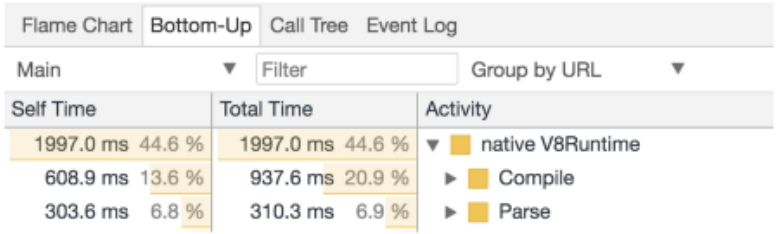
MOST EXPENSIVE ASSET

JavaScript bytes !== JPEG bytes

~170KB ~170KB

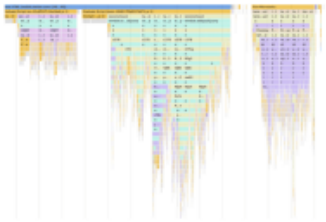
Network Transmission

Resource Processing



~2s in Parse/Compile

0.064s in Image Decode



~1.5s in Execution

0.028s in Rasterize Paint

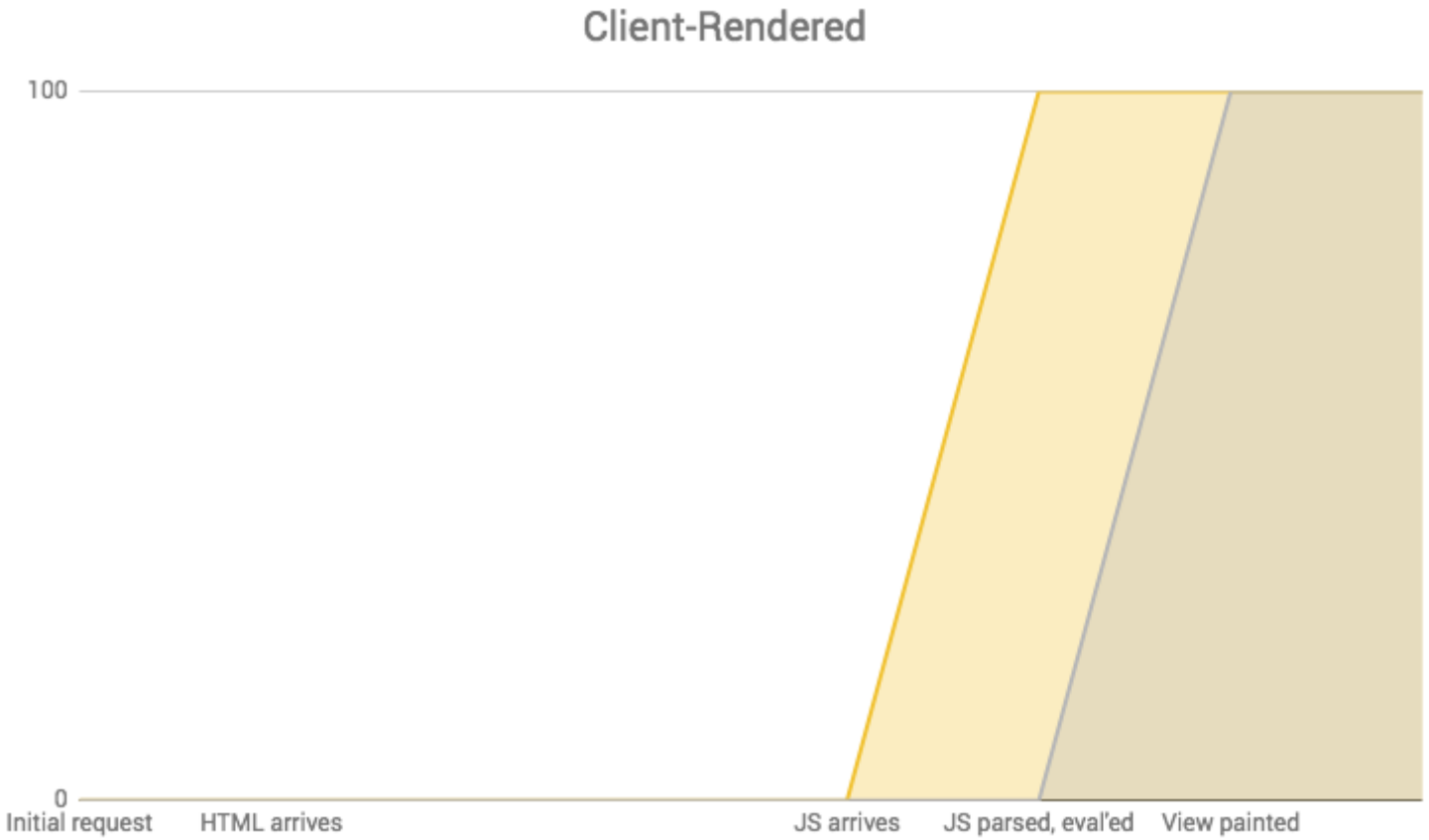
@addyosmani - 170KB of (compressed) JS vs. JPEG bytes over a slow 3G network on a Moto G4. JS needing parsed is even larger once decompressed.

https://medium.com/dev-channel/the-cost-of-javascript-84009f51e99e

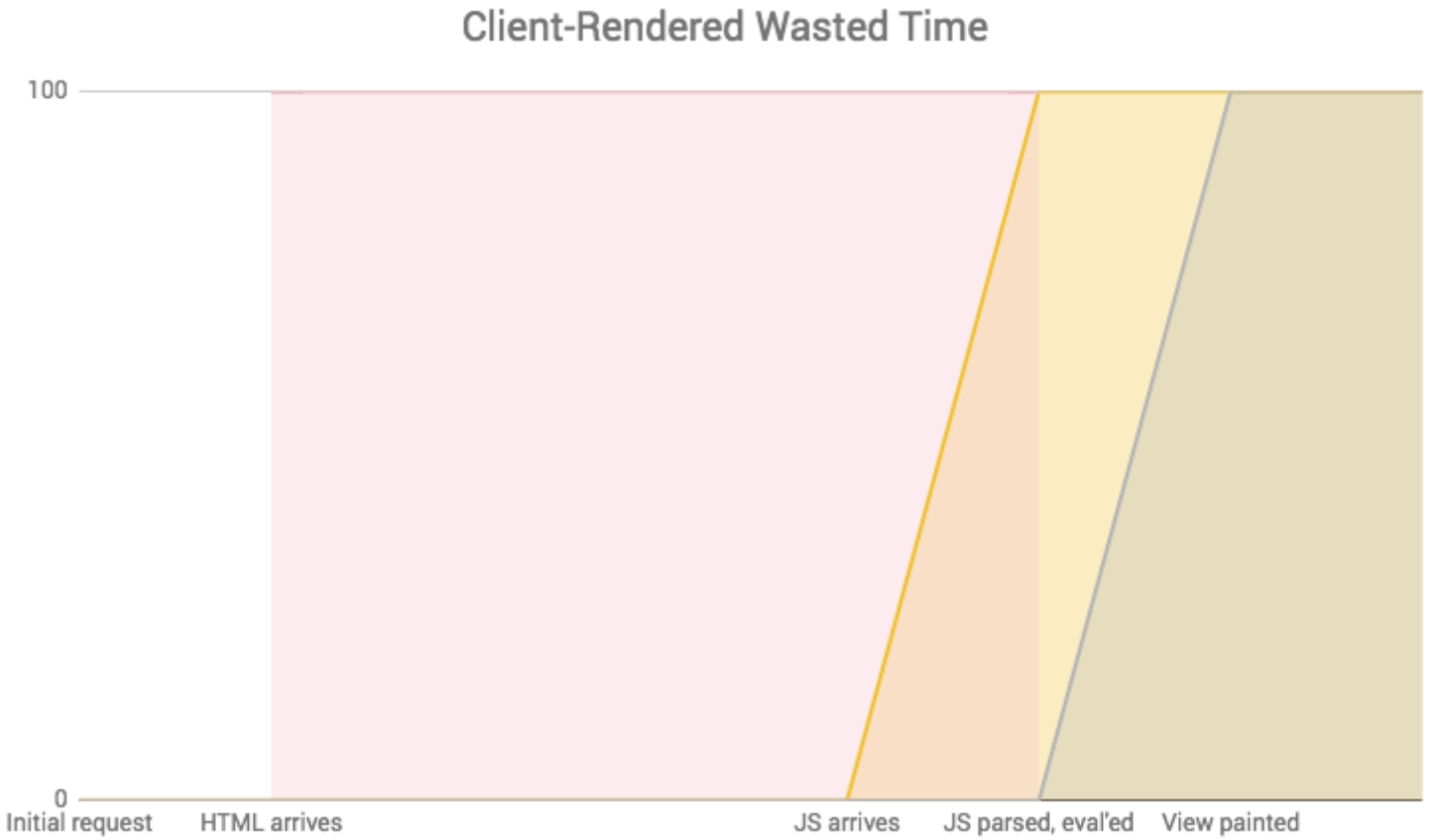
TL;DR: Ship less code

- less code = less load + less parse/compile
- holy grail = prioritize only what's needed in view

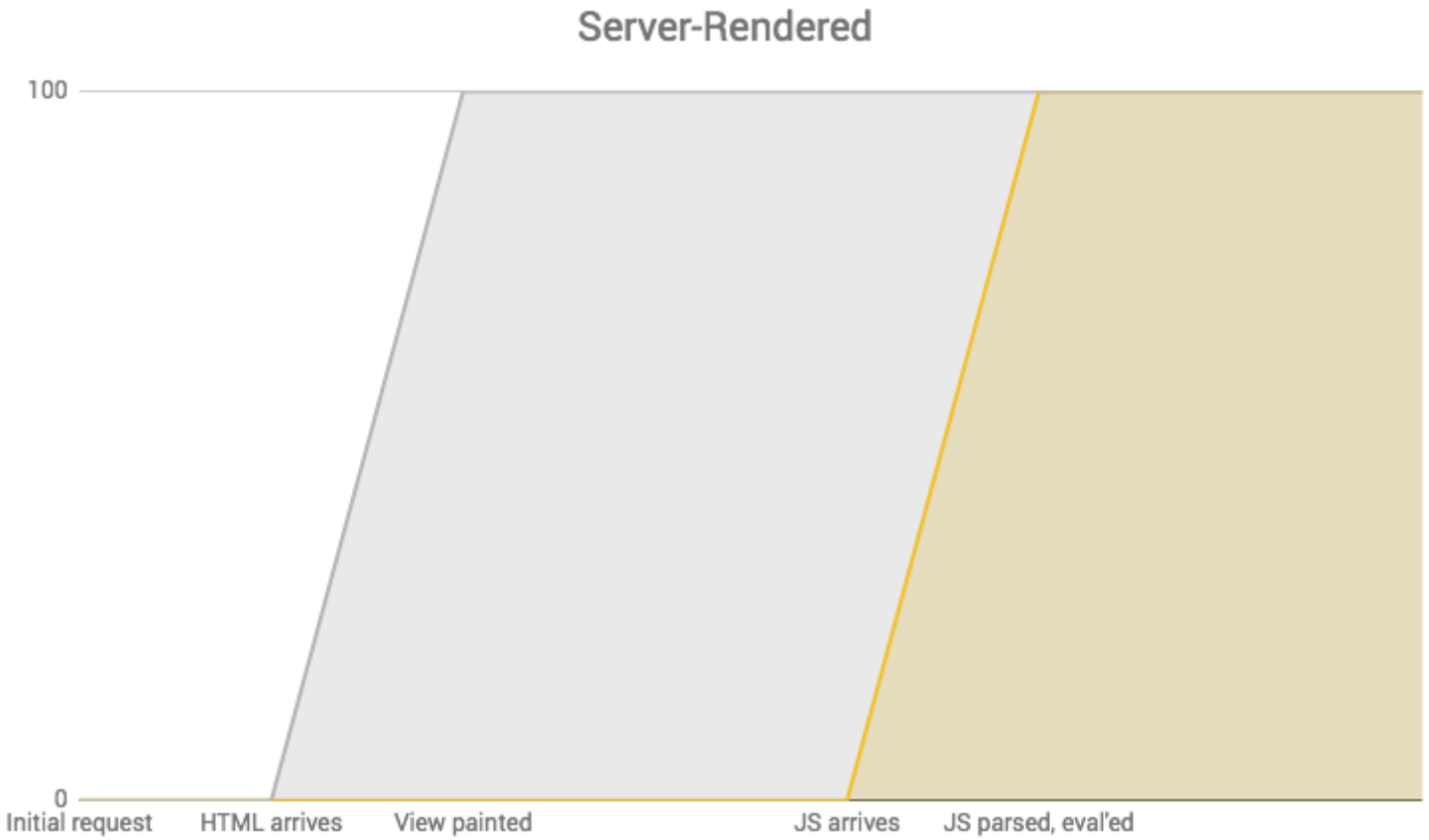
Client vs Server vs Progressive Rendering



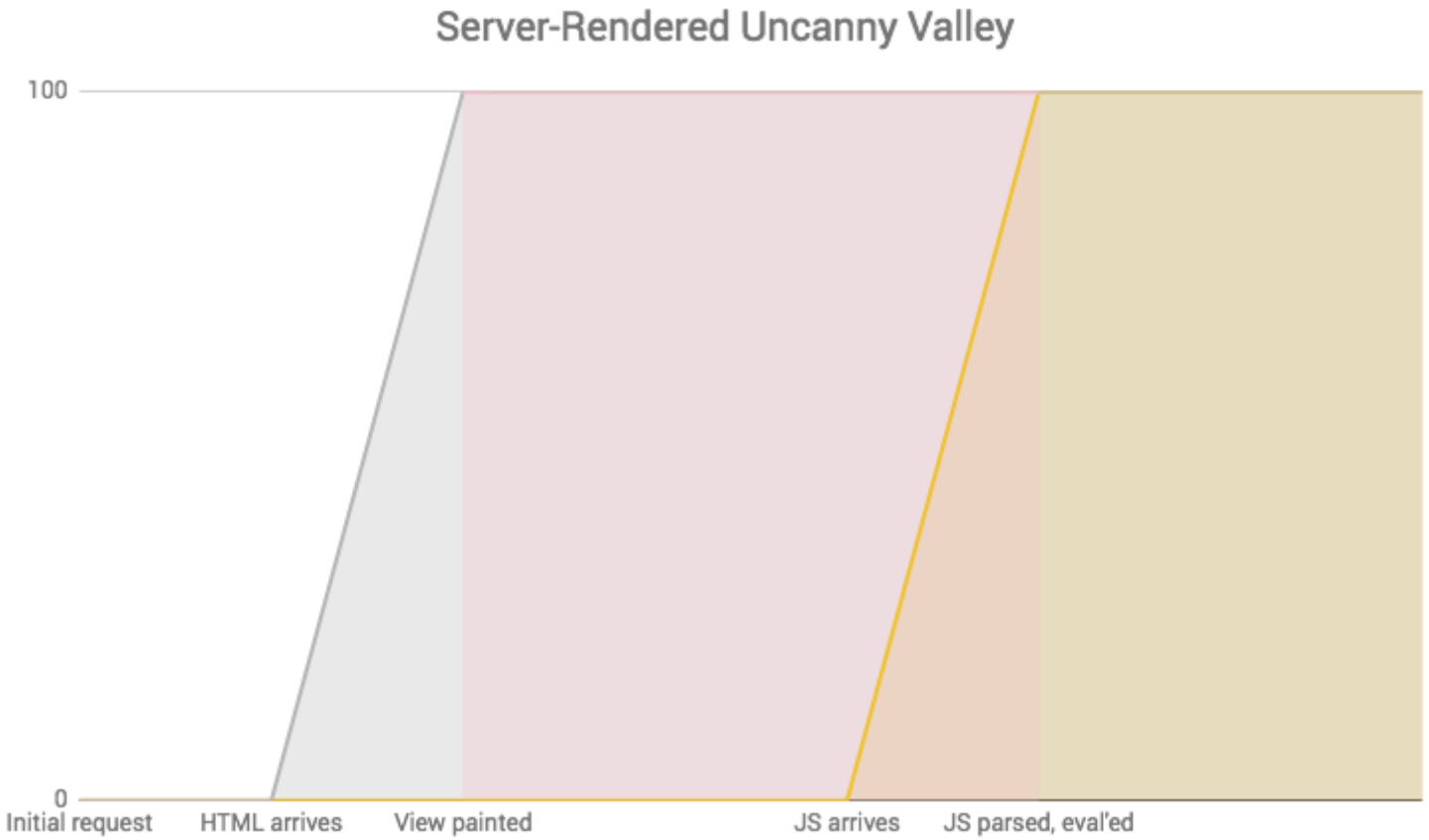
Client vs Server vs Progressive Rendering



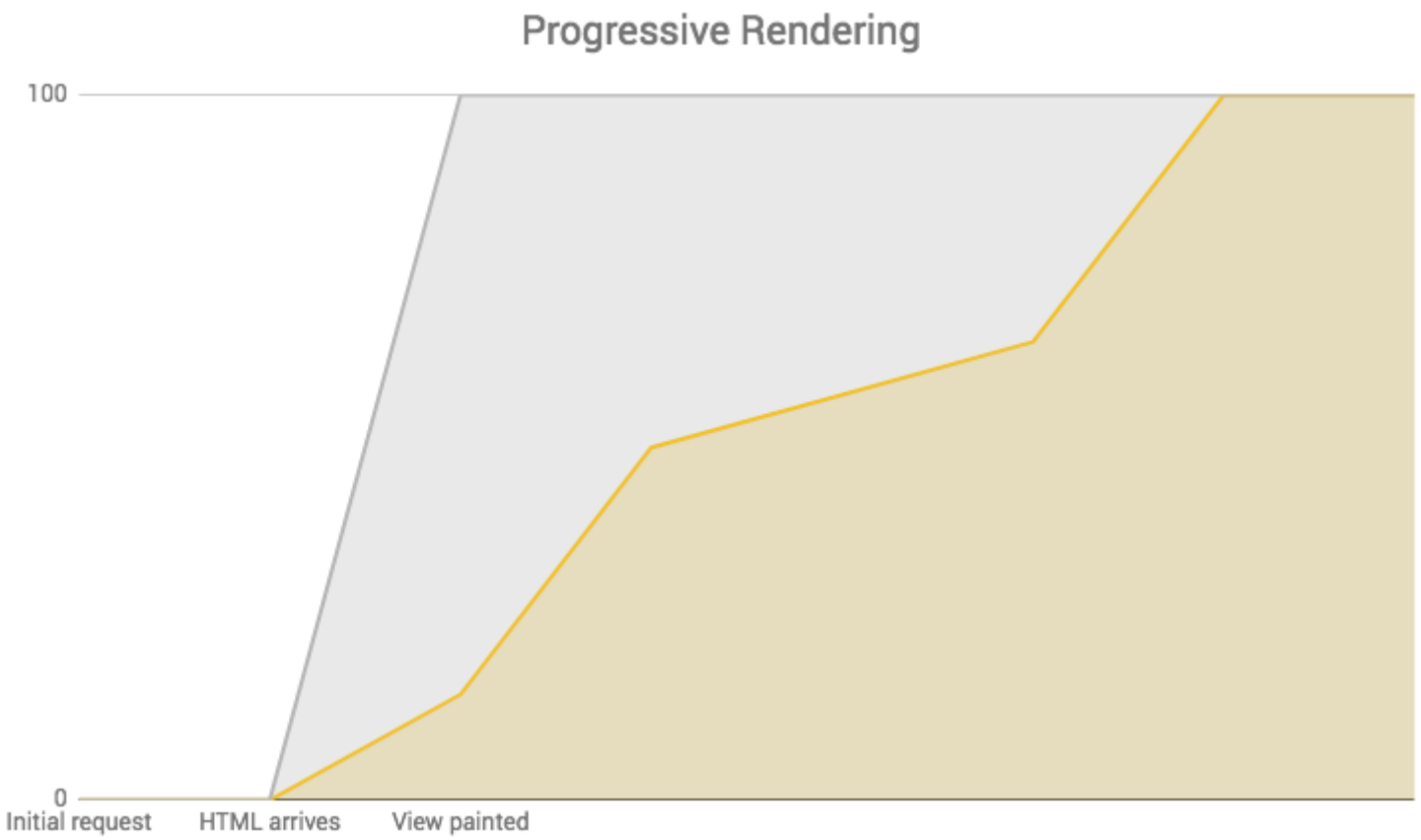
Client vs Server vs Progressive Rendering



Client vs Server vs Progressive Rendering



Client vs Server vs Progressive Rendering



Inspired by https://twitter.com/aerotwist/status/729712502943174657

Optimizing Time to Interactive

- **Analyze your loads and bundles! Don't over-optimize!**
- **Only ship what's immediately needed** - use code splitting, pre-caching, and lazy loading.
- Migrate to **HTTP2** for concurrent requests and header compression.
- **Minify** to speed up both download and parse/compile.
- **Compress** with gzip or brotli.
- **Remove unused code** with tree shaking and using module imports effectively.

Module Imports

```
// Big
import _ from 'lodash';
_.isEmpty({});

// Little
import isEmpty from 'lodash/isEmpty';
isEmpty({})

// Big
import moment from 'moment';

// Little
import addMinutes from 'date-fns/addMinutes';
```

The Cost of Unnecessary Transpiling

Version	Size (minified)	Size (minified + gzipped)	Parse/eval time (avg)
ES2015+	80K	**21K**	**172ms**
ES5	175K	**43K**	**367ms**

https://philipwalton.com/articles/deploying-es2015-code-in-production-today/

> *Most of your time is spent using the app, not waiting to load.*

Devin Villegas, Netflix senior dev ops engineer

Optimizing Responsiveness

- **Don't block the main thread!**
- Avoid memory leaks - garbage collection can pause execution
- Avoid long-running JS - chunk into smaller pieces with `requestAnimationFrame()` or `requestIdleCallback()` for scheduling
- Use up-to-date frameworks that prioritize user input (like React Fiber starting in React v16.0)

https://medium.com/dev-channel/the-cost-of-javascript-84009f51e99e
https://philipwalton.com/articles/why-web-developers-need-to-care-about-interactivity/

HOUSTON'S BAGGAGE CLAIM COMPLAINTS

http://www.nytimes.com/2012/08/19/opinion/sunday/why-waiting-in-line-is-torture.html

> *Are you better off making the site load faster or ensuring that users complete their tasks?*

Christine Perfetti, The Truth About Download Time 2006


THANKS!
Slides, resources, and more at bit.ly/siaspeaks