Introductory Note

Every so often we see twitter rants about progressive enhancnement versus using Javascript for everything... after all we can do most of what CSS can with Javascript and the CSS Object Model, right?

The latest iteration of this debate has been between Alex Russell who advocates for smaller JS payloads, lazy loading and progressive enhancement versus the people saying that we shouldn't worry about javascript being available because all modern browsers are evergreen and people wouldn't think about disabling Javascript.

These are some random toughts and ideas about the subject

Do we still need progressive enhancement?

Rather than get into the debate between Javascript or not Javascript I'll try to go back to the core of what it means to progressively enhance a page or an app, contrast it with graceful degradation, present some of the modern alternatives to the "no Javascript" and explain why this is important (to me)

Image from Understanding Progressive Enhancement (A List Apart) by Aaron Gustafson

At the core of the web we have content. Whether you're trying to sell me something with your shinny web application or giving me a summary of War and Peace you're providing me with content that I will interact with. It doesn't matter how ugly the content looks, we just want it to be there.

Once we have our content built with Semantic (X)HTML we can build upon it. Now we can add all the awesome things we can do with CSS: Flexbox, Grids, animations, transitions and many other things that can be done in CSS but not in HTML.

Only then we should think about those beautiful D3 visualizations we've worked on for the last three months. **None of these things will replace the content, only enhance it**. If the user has disabled images or if Javascript download times out we wil still have the content.

The other side of the progressive enhancement is graceful degradation where we use modern browsers as our development baseline and only provide basic fixes for older browsers (which are defined differently in different frameworks and even different applications).

So we'll take it for granted that Javascript is available on all browsers. That doesn't mean we should Javascript all things. We need to consider progressive enhancement in light of where our next users will come from, the type of devices and network connectivity that the new users have access to.

There are issues of mobile itself. The latency inherent in the mobile networks, the fact that the users expect the site to be fast even when the mobile browser has to wait for cores to start up when in powered down mode, then wait for the wireless transceiver to power up, send the signal and receive data from the network (including large amounts of Javascript and images) and only then it can render the content of the page.

I'm already picturing the comments that this will not affect your users. You're right, it won't affect your users most of the time. We've grown used to the bandwidth we share in urban areas and forget, for a minute, that not all people share the same bandwidth... even in the best days I get LTE or 3G connection which will slow down content rendering living in the San Francisco Bay Area of California.

There are other issues in mobile that impact the performance of our mobile hardware, especially the lower end devices. The way that the hardware on the devices are built and the types of cores built into the mobile CPUs can all affect the performance of the devices.

The best explanation I've heard so far is from Alex Russell and delivered at the Chrome Dev Summit earlier this year.

Particularly in emerging economies network access can cost a significant portion of a person's salary. That changes the people access content and should dictate at least part of how we develop our content.

We can no longer assume that building applications as monolithic structures will work for all our users. We need to test our applications and content in the actual devices that we're targeting or, as a second alternative, use webpagetest and test from the location where you expect your new customers to be.

Bruce Lawson from Opera presents the other side of the picture. Where are people coming (geographically), How can developers make it easier, and more afordable, for users to access your applications.

He also talks about proxy browsers and gives design best practices and what will and won't work in a proxy browser. My favorite part of Bruce's presentation is ***Bruce's Law of Smartness™***

It doesn't matter how smart your phone is if your network is dumb

> Let's make sure we keep the World Wide Web rather than the Wealthy Westerners' Web (H/T, again, to Bruce Lawson).