Advanced Page Speed Optimization

Nobody Likes a Slow-loading Website

- The Mobile Web Experience Absolutely Sucks
- Mainly a mobile issue
  - Processing power vs desktop / tablet / laptop
  - Attention span
- Tremendous impact on marketing conversions, bounce rate

Google / SOASTA Deep Machine Learning

- Analysis of 900,000 mobile ads' landing pages spanning 126 countries
- Tested impact of page loading speed on conversions and bounce rates

- "Recently, we trained a deep neural network—a computer system modeled on the human brain and nervous system—with a large set of bounce rate and conversions data. The neural net, which had a 90% prediction accuracy, found that as page load time goes from one second to seven seconds, the probability of a mobile site visitor bouncing increases 113%. Similarly, as the number of elements—text, titles, images—on a page goes from 400 to 6,000, the probability of conversion drops 95%."

Key Results From Google / SOASTA

- "The average time it takes to fully load a mobile landing page is 22 seconds"
- "53% of visits are abandoned if a mobile site takes longer than three seconds to load"
- "For 70% of the pages we analyzed, it took nearly seven seconds for the visual content above the fold to display on the screen"
- "We found that 70% of pages were over 1MB, 36% over 2MB and 12% over 4MB"

Google's PageSpeed Insights Rules Links

- Speed Rules
  - Avoid landing page redirects
  - Enable compression
  - Improve server response time
  - Leverage browser caching
  - Minify resources
  - Optimize images
  - Optimize CSS Delivery *
  - Prioritize visible content *
  - Remove render-blocking JavaScript *
  - Use asynchronous scripts *

- Usability Rules
  - Avoid plugins
  - Configure the viewport
  - Size content to viewport
  - Size tap targets appropriately
  - Use legible font sizes

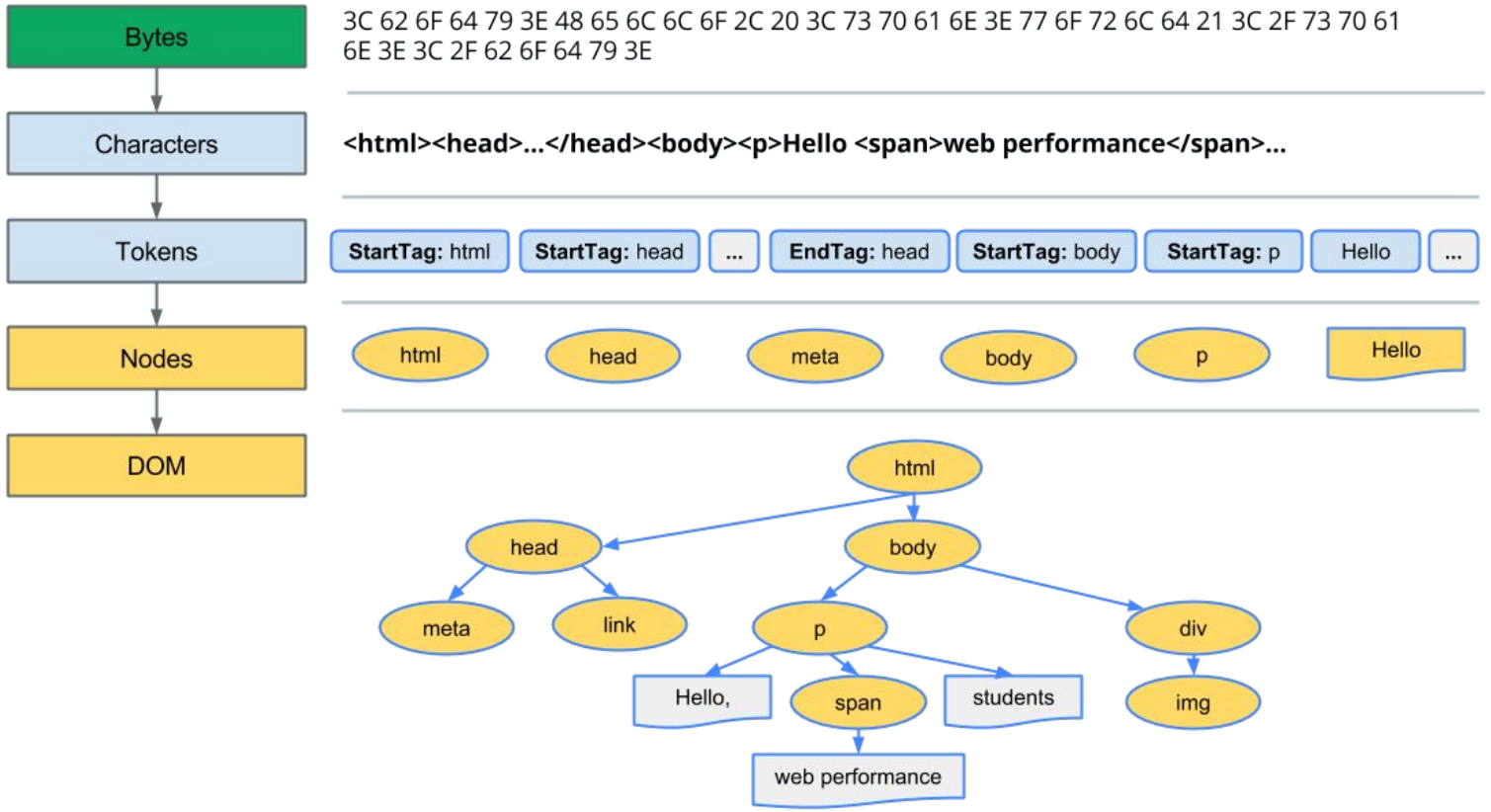
- * Denotes Presentation Focus

Understanding How Browsers Render Web Pages

Browser Rendering Steps

1. Process HTML markup and build the DOM tree.
2. Process CSS markup and build the CSSOM tree.
3. Combine the DOM and CSSOM into a render tree.
4. Run layout on the render tree to compute geometry of each node.
5. Paint the individual nodes to the screen.

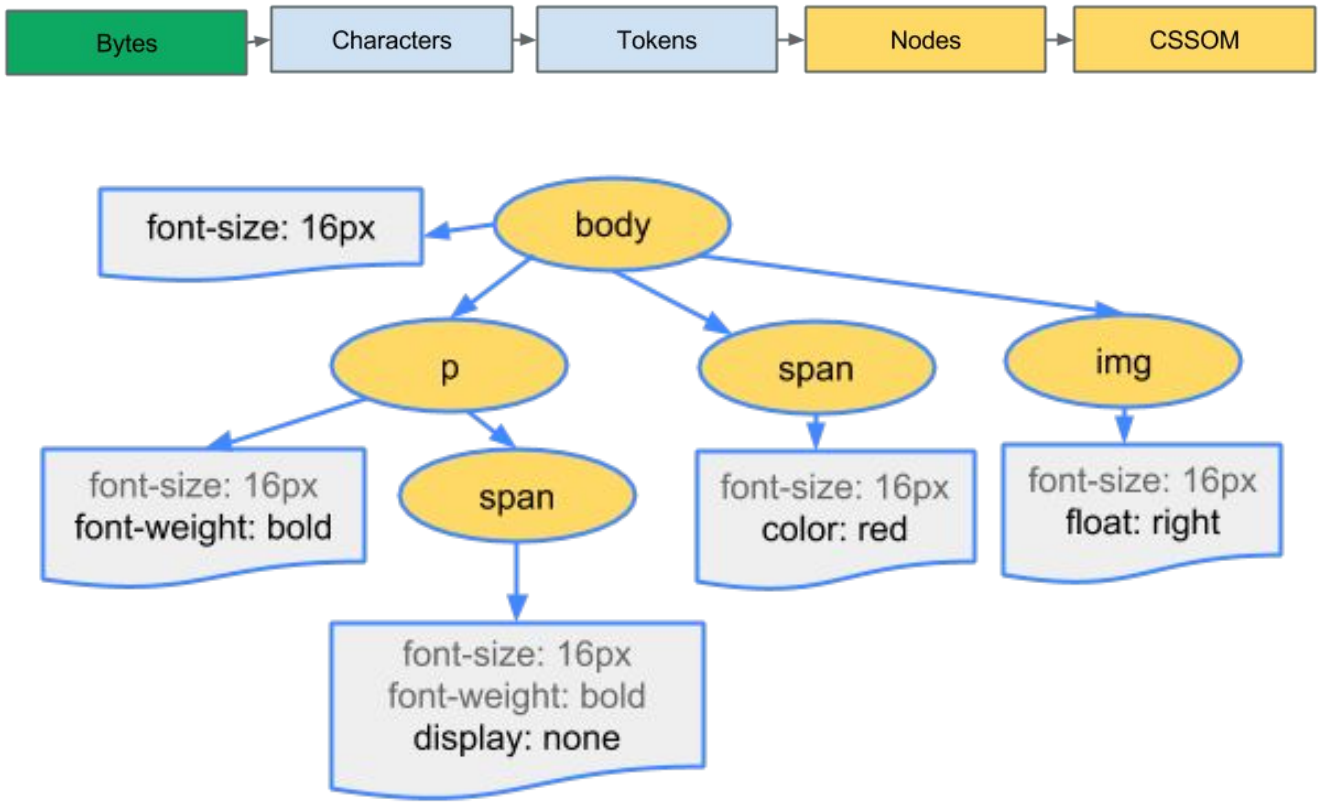
Building the DOM Tree



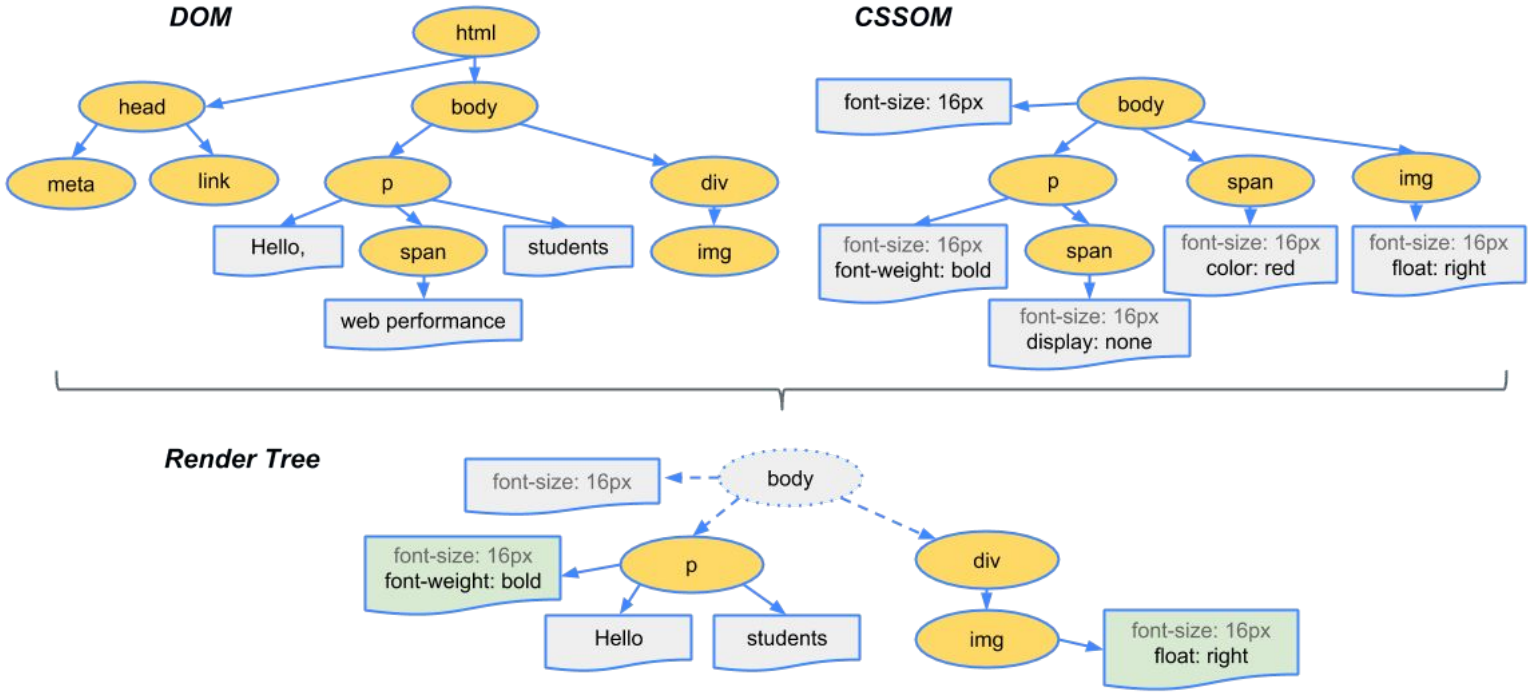
Building the DOM Tree (cont.)

- Raw HTML bytes converted to characters (utf-8)
- Character strings converted to tokens (i.e. tags and encapsulated text)
  - Each token type defines meaning/purpose, sets predefined rules
  - For MUCH more info: https://www.w3.org/TR/html5/syntax.html#parsing-main-inhtml
- Tokens are converted into objects with properties and rules
- DOM tree is constructed using HTML parent-child hierarchy structure (i.e. nested elements)

Building the CSSOM Tree



Building the Render Tree



Build Geometric Layout and Painting Result

- Layout (Reflow) Stage
  - Browser traverses Render Tree from root
  - Computes node geometry and placement
- Paint Stage
  - Nodes rendered on screen

- "If either the DOM or CSSOM were modified, you would have to repeat the process in order to figure out which pixels would need to be re-rendered on the screen."

Optimizing Performance With Browser Rendering in Mind

The Critical Rendering Path

- It is the series of events that must take place to render (display) the initial view of a webpage.
- Example: get html > get resources > parse > display webpage
- It gives you the power to make a large webpage with many resources load faster than a small webpage with few resources.

Varvy Example

- browser downloads the html file
- browser reads the html and sees that there are one css file, one javascript file and one image
- browser starts downloading the image
- browser decides it can not display the webpage without first getting the css and javascript
- browser downloads the CSS file and reads it to make sure nothing else is being called
- browser decides it still can not display the webpage yet until it has the javascript
- browser downloads the javascript file and reads it to make sure nothing else is being called
- Browser now decides it can display the webpage

Eliminating Render Blocking CSS and JS

- Pages cannot be rendered until:
  - All CSS files have been parsed into the CSSOM tree
  - JavaScript-dependent content has been processed

What is Above the Fold Content?

- Landing page content immediately visible in viewport upon page load

Optimizing Above the Fold Content

- Make sure your pages are calling stylesheets before JavaScripts.
- No more than one external stylesheet
- Don't use @import
- Avoid (multiple) custom fonts, JavaScript Plugins
- Inline "small" Critical CSS using style tags, not in-element styling
- Separate JavaScript into two groups: Render-dependent and Post-load functionality (click events, etc.)
  - Async scripts execute immediately after it is downloaded
  - Deferred scripts immediate begin downloading, but execute after page load
- Base 64 encode necessary images
- Optimize images
- Lazy-load / defer images

Remaining Questions

- How small is “small” inline CSS?
- Should critical CSS cover all landing pages and viewports?
- Deferred Javascript – defer attribute vs code solution?

Sweet Web Tools

- Varvy SEO Tools – Most thorough SEO/Performance toolset and documentation available
- PageSpeed Tools – Google's PageSpeed insights tool
- TestMySite with Google – Thorough SEO and Performance analysis with detailed report
- Critical – Gulp.js extension that automatically separates and inlines Critical CSS
- Lazy-Loading Images – Article and demo of lazy-loading technique
- Fix analytics.js Browser Caching Error

Source Articles

- https://www.thinkwithgoogle.com/marketing-resources/experience-design/mobile-page-speed-load-time/
- https://www.thinkwithgoogle.com/marketing-resources/data-measurement/mobile-page-speed-new-industry-benchmarks/
- https://www.soasta.com/blog/google-machine-learning-ecommerce-research/
- https://www.soasta.com/blog/mobile-web-performance-monitoring-conversion-rate/
- https://developers.google.com/web/fundamentals/performance/critical-rendering-path/constructing-the-object-model

Source Articles

- https://developers.google.com/web/fundamentals/performance/critical-rendering-path/render-tree-construction
- https://www.section.io/blog/page-load-time-bounce-rate/
- https://www.sitepoint.com/how-and-why-you-should-inline-your-critical-css/