Vishwakarma, Anand A.

From: Vishwakarma, Anand A.
Sent: Tuesday, November 20, 2018 12:20 PM
To: Negi Kumar, Naresh; Sannamanjegowda, M.
Cc: Chandrashekar, G. S.
Subject: RE: POC task

Hi Naresh,

I have found some solution for below listed task. Please have look into it and let me know if you have any concern.

FRONTEND TASK	REQUIREMENTS	
Defer offscreen images	- On supported devices i.e. iPhone, Android, Desktop, determine which images load below the fold and apply lazy-loading wherever possible. - "Bonus points" for combining lazy-loading with "next-gen" image formats (see below)	As we discussed, W For IE browser, hav
Serve images in next-gen formats	- Update images using <picture>tag with <srcset>to WebP with specified fallbacks. - Use feature detection library like Modernizr (already in use) for detecting WebP support using CSS - This feature would be highly dependent the dynamic imaging service's ability to convert images to WebP format for product images	WEBP image format s polyfill. Below are the 1) <img src="image.we 2) <picture> <source srcset="i <img src="images/ </picture> 3) CSS fallback Modernizr : detecti
Optimize CSS	- Remove unused rules from stylesheets to reduce unnecessary bytes consumed by network activity. - Use a tool such as Coverage to determine how much to begin trimming	CSS Lint is a tool to he http://csslint.net/
Reduce the time spent parsing, compiling and executing JS	- Pre-load elements –make sure elements in the <head> are pre-loaded, before the visitor sees anything in browser - Use preload, prefetch, and preconnect to inform the browser ahead of future network requests (e.g CDN domains like CoreMedia, Monetate, cQuotient, Bazaarvoice, and Telium) - Trim dependencies-analyze Bundles (Browserify), determine what is needed and what is "cruft" within those bundles - Coverage Analyzer -determine which lines of code are being used. - Modular Libraries –require.js by function vs. Entire library - Split JS Bundles –only load the JS you need e.g. if loading JS for another page, lazy-load it later in the pipeline	-
Minify JavaScript	- Modify build scripts - Use minified versions of 3rdparty JS - Consider one or more bundled script payloads to reduce network requests	-
Eliminate Render-blocking Resources	- For critical scripts, consider inlining them in your HTML - For non-critical scripts, consider marking them with the asyncor deferattributes - For stylesheets, consider splitting up your styles into different files, organized by media query, and then adding a mediaattribute to each stylesheet link - When loading a page, the browser only blocks the first paint to retrieve the stylesheets that match the user's device - For non-critical HTML imports, mark them with the asyncattribute. As a general rule, asyncshould be used with HTML imports as much as possible	-

Defer offscreen images	- Lazy-load main navigation images upon user interaction or do not load them on mobile	-
Optimize delivery of product images	- Reduce page weight by delivering images at the most appropriate size for the current viewport/device/pixel density by using a responsive image approach – Use DIS to rescale images on product tiles	-
UX/UI improvements	- Review client side code for rendering bottlenecks in CSS and JavaScript (e.g. use GPU accelerated animations, defer component initialization) - Review UI interaction patterns for best practices (e.g. jarring navigation interactions, page jumps, and scrolling performance)	-
Set a Performance Budget	- Access and Determine the limit - Set up notifiications/Dashboard	-
Monitor performance continuously	- Share examples for dashboard and heartbeat monitor	-
Evaluate, choose and use the right tools to find and fix performance issues	- Determine the best tools to use moving forward	1) developers.g The PageSpee on both mobi that page ma

Regards,
Anand.

From: Negi Kumar, Naresh
Sent: Tuesday, November 20, 2018 10:52 AM
To: Sannamanjegowda, M. <m.sannamanjegowda@accenture.com>; Vishwakarma, Anand A. <anand.a.vishwakarma@accenture.com>
Cc: Chandrashekar, G. S. <g.s.chandrashekar@accenture.com>
Subject: RE: POC task

Thanks Manasa,

For “Defer offscreen images” do google “browser intersection observer api”. You will get an idea.

Also Please share your input over WEBP Support.

- Use feature detection library like Modernizr (already in use) for detecting WebP support using CSS

This feature would be highly dependent the dynamic imaging service’s ability to convert images to WebP format for product images

Thanks &Regards
Naresh Negi

Accenture

From: Sannamanjegowda, M.
Sent: Monday, November 19, 2018 5:27 PM
To: Negi Kumar, Naresh <naresh.negi.kumar@accenture.com>; Anand, Kinner G. <kinner.g.anand@accenture.com>
Cc: Chandrashekar, G. S. <g.s.chandrashekar@accenture.com>
Subject: RE: POC task

Hi Naresh,

I went through the task listed below and understood the task, except "Defer offscreen images". We have some existing solution for other points, we need to work on that.

Thanks & Regards,
Manasa K.S

From: Negi Kumar, Naresh
Sent: Friday, November 16, 2018 4:01 PM
To: Anand, Kinner G. <kinner.g.anand@accenture.com>; Sannamanjegowda, M. <m.sannamanjegowda@accenture.com>
Cc: Chandrashekar, G. S. <g.s.chandrashekar@accenture.com>
Subject: POC task

Hello Anand and Mansa,
Below is the list of FRONTEND TASK, which are a key requirement for the future project (Pandora). I highlighted the things which are not available in SFCC.

Please do some POC over it and create some generic code base, so that could be use in other project as well (just plug and play type).

detecting WebP support using CSS: Anand already did basic POC over it, please go deep dive in to this and make it compatible for all the browser (with fallback option for the browser, which doesn't support WEBP).

FRONTEND TASK	REQUIREMENTS
Defer offscreen images	- On supported devices i.e. iPhone, Android, Desktop, determine which images load below the fold and apply lazy-loading wherever possible. - "Bonus points" for combining lazy-loading with "next-gen" image formats (see below)
Serve images in next-gen formats	- Update images using <picture>tag with <srcset>to WebP with specified fallbacks. - Use feature detection library like Modernizr (already in use) for detecting WebP support using CSS - This feature would be highly dependent the dynamic imaging service's ability to convert images to WebP format for product images
Optimize CSS	- Remove unused rules from stylesheets to reduce unnecessary bytes consumed by network activity. - Use a tool such as Coverage to determine how much to begin trimming
Reduce the time spent parsing, compiling and executing JS	- Pre-load elements –make sure elements in the <head> are pre-loaded, before the visitor sees anything in browser - Use preload, prefetch, and preconnect to inform the browser ahead of future network requests (e.g CDN domains like CoreMedia, Monetate, cQuotient, Bazaarvoice, and Telium)

	- Trim dependencies-analyze Bundles (Browserify), determine what is needed and what is "cruft" within those bundles - Coverage Analyzer -determine which lines of code are being used. - Modular Libraries –require.js by function vs. Entire library - Split JS Bundles –only load the JS you need e.g. if loading JS for another page, lazy-load it later in the pipeline
Minify JavaScript	- Modify build scripts - Use minified versions of 3rdparty JS - Consider one or more bundled script payloads to reduce network requests
Eliminate Render-blocking Resources	- For critical scripts, consider inlining them in your HTML - For non-critical scripts, consider marking them with the asyncor deferattributes - For stylesheets, consider splitting up your styles into different files, organized by media query, and then adding a mediaattribute to each stylesheet link - When loading a page, the browser only blocks the first paint to retrieve the stylesheets that match the user's device - For non-critical HTML imports, mark them with the asyncattribute. As a general rule, asyncshould be used with HTML imports as much as possible
Defer offscreen images	- Lazy-load main navigation images upon user interaction or do not load them on mobile
Optimize delivery of product images	- Reduce page weight by delivering images at the most appropriate size for the current viewport/device/pixel density by using a responsive image approach – Use DIS to rescale images on product tiles
UX/UI improvements	- Review client side code for rendering bottlenecks in CSS and JavaScript (e.g. use GPU accelerated animations, defer component initialization) - Review UI interaction patterns for best practices (e.g. jarring navigation interactions, page jumps, and scrolling performance)
Set a Performance Budget	- Access and Determine the limit - Set up notifiications/Dashboard
Monitor performance continuously	- Share examples for dashboard and heartbeat monitor
Evaluate, choose and use the right tools to find and fix performance issues	- Determine the best tools to use moving forward

Thanks &Regards
Naresh Negi

Accenture