UCSD CSE 190 A00 WEB APP PERFORMANCE AND MONITORING SPRING 2020 **PUBLIC SITE PERFORMANCE REMEDIATION** By Gabriela Shirley



Mirror UCSD Homepage to my Github Pages

Moving only the assets of the homepage to my gabrielashirley.github.io repository, it takes up 5.9MB space to download instead of 72.1MB when I tried to mirror the entire UCSD site. The link to the original site that I mirrored is here: <u>https://gabrielashirley.github.io/original/</u>.



Taking a baseline reading from Lighthouse:

It does not show any significant difference compared to when I tested the real site (https://ucsd.edu/) on the second picture. However, my first guess on the 3 points better in performance, with every smaller (means better) number under the metrics when I served the homepage on Github Pages has something to do with how good is Github Pages' backend on serving static sites. As discussed in this site. https://www.savjee.be/2017/10/Static-website-hosting-who-is-fastest/, Github Pages wins as the fastest and that everyone performs fairly consistently compared to other hosting services such as Netlify, Amazon S3, CloudFront, Firebase Hosting, and Google Cloud.

Taking a baseline reading from WebPageTest:

Web Pa https://gabrie From: Los Ang 5/19/2020, 9:00	elashirle geles, CA 8:48 PM	Perfo ey.githuk - Chrome	D.io/original	e Te: /	st for			Security score	First E Time	A A Byte Keep-al Enabled	ive Com Trans	A press Con	npress ges Cac stati cont	he E c u ent	ffective se of CDN
ummary Do Tester: webpag First View onl Test runs: 3 Re-run the te	etails getest2-64 y st	Performa 4.183.41.1	ance Review	Cont	ent Break	down Doi	mains Pro	cessing Bre	spee	Screens	shot l	lmage Ana <u>Raw</u>	alysis & F / <u>page data</u> <u>Export H</u>	Request View JS0 - <u>Raw ob</u> TTP Arch <u>View</u>	Map & ON result <u>oject data</u> <u>ive (.har)</u> Test Log
							Web Vitals		Doc	ument Com	plete		Fully Lo	aded	
	First View (Run 2)StartSpeed LotLargest Contentful HeroCumulative LargestTotal Blocking TimeSpeed InSpeed IndexSpeed Largest Contentful PaintCumulative Largest Contentful PaintTotal Blocking TimeSpeed InSpeed InSpeed InSpeed InSpeed InLargest Contentful PaintTotal Blocking TimeSpeed InSinth<														
First View (<u>Run 2</u>)	First View (Run 2) 0.135s 3.200s 3.025s 3.806s 4.300s 3.128s 0.054 ≥ 0.383s 8.789s 131 4,317 KB 11.169s 156 5,526 KB \$														
Web Pa https://ucsd. From: Los Ang 5/19/2020, 8:11	First View (Run 2) 0.135s 3.200s 3.025s 3.806s 4.300s 3.128s 0.054 ≥ 0.383s 8.789s 131 4.317 KB 11.169s 156 5.526 KB \$														
ummary Do Tester: webpag First View only Test runs: 3 Re-run the te	e tails getest-1-6 y st	Performa 64.183.41.	ance Review 10	Cont	ent Break	down Do	mains Pro	cessing Bro	eakdowr	n Screen:	shot	Image An <u>Rav</u>	alysis ଔ v page data Export H	Request <u>View JS</u> - <u>Raw ol</u> TTP Arch <u>View</u>	Map ON result bject data hive (.har) r Test Log
				Perfo	ormano	ce Resul	ts (Media	n Run -	Spee	dIndex)					
					I	,	Web Vitale		Doci	iment Comr	alete		Fully Lo	hahe	

							Web Vitals		Docu	iment Comp	olete		Fully Loa	ded	
	First Byte	Start Render	<u>First</u> Contentful <u>Paint</u>	<u>Speed</u> Index	<u>Last</u> Painted <u>Hero</u>	<u>Largest</u> <u>Contentful</u> <u>Paint</u>	<u>Cumulative</u> Layout Shift	<u>Total</u> <u>Blocking</u> <u>Time</u>	Time	Requests	Bytes In	Time	Requests	Bytes In	Cost
First View (<u>Run 2</u>)	0.309s	6.000s	5.890s	6.511s	6.700s	5.970s	0.055	≥ 0.269s	11.289s	129	5,595 KB	13.389s	148	6,607 KB	<u>\$\$\$\$\$</u>

Comparing the test results of my mirror and the real site, I observed that the total opposite score of Compress Transfer and Effective use of Content Delivery Network (CDN) (Mirror: A and ✓ vs Real: F and X) which in other words means the way the we fetch compressible static assets does make a difference. Since Github automatically compresses required files into GZip if the browser supports it, whereas UCSD Server seems to not turn the feature on (or maybe it does not support it), it helps saving time on the resource load times as well as slow server response times. One extra thing that might help improve the Time to First Byte (TTFB) of the real site is the use of CDN like Cloudflare, where it will manage the server routing to the nearest ones from the user, thus reducing latency. Currently only the third party assets use CDN like Google (including YouTube), Facebook, Cloudflare, etc.

Start render time is suggested to be around 1-2s. However, I noticed that both my mirror and the real site have some kind of render blocking due to several CSS files which results in having start render in 3.2s and 6.0s respectively.



Waterfall View

4 Gabriela Shirley

Optimization

So to begin with my optimization attempts, I would take my mirror test results as my baseline from now on.

1) Eliminate render-blocking (apply *defer* and *rel=preload*) and add <noscript> fallback

Based on my observation, these resources can be deferred for later load and execution or removed thus increasing start render time. Taken from class notion notes, scripts with the defer attribute execute after HTML parsing is completely finished, but before the DOMContentLoaded event. defer guarantees scripts will be executed in the order they appear in the HTML and will not block the parser.

Console What's New Coverage X				
Pertunction * 🧕 🛇 👱 URL filter 🛛 🗚 🔻 🕅 Conte	ent scripts			
URL	Type	Total Bytes	Unused Bytes	Usage Visualization
https://gabrielashinley.github.lo/optimized/_resources/css/vendor/brix_sans.css	CSS	2 113 450	2 113 450100.0 %	
https://connect.facebook.net/signals/config/498850027599111?v=2.9.18&r=stable	JS (per function)	528 671	454 950 88.1 %	
https://gabrielashirley.github.io/optimized/_resources/css/styles.css	CSS	170 064	145 801 85.7 %	
https://gabrielashirley.github.io/optimized/_resources/css/custom.css	CSS	107 191	96 353 89.9 %	
https://gabrielashirley.github.io/optimized/_rescurces/css/vendor/animate.css	CSS	72 253	71 966 99.6 %	
https://connect.facebook.net/en_US/fbevents.js	JS (per function)	134 009	65 788 49.1 %	
https://odnjs.cloudflare.com/ajax/libs/gsap/2.0.2/TweenMax.min.js	JS (per function)	116 074	50 748 43.7 %	_
https://www.googletagmanager.com/gtm.js?id=GTM-K826VG	JS (per function)	347 572	49 315 14.2 %	
https://s.ytimg.com/yts/jsbin/www-widgetapi-vfiZhIU7/www-widgetapi.js	JS (per function)	68 782	44 361 64.5 %	
https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js	JS (per function)	95 992	38 959 40.6 %	
https://gabrielashirley.github.io/optimized/_resources/js/vendor/slick.min.js	JS (per function)	41 913	38 703 92.3 %	
https://gabrielashirley.github.io/optimized/_resources/css/vendor/font-awesome.css	CSS	35 243	34 962 99.2 %	
https://gabrielashirley.github.io/optimized/_resources/bootstrap/jav /bootstrap.min.js	JS (per function)	37 045	30 385 82.0 %	
https://gabrielashirley.github.io/optimized/_resources/css/custom-img.css	CSS	19 233	17 323 90.1 %	
https://www.google-analytics.com/analytics.js	JS (per function)	45 892	14 941 32.6 %	
https://gabrielashirley.github.io/optimized/_resources/js/myscripts.js	JS (per function)	27 673	14 167 51.2 %	
https://gabrielashirley.github.io/optimized/_resources/js/vendor/jquery.easings.min.js	JS (per function)	12 927	9 769 75.6 %	
https://gabrielashirley.github.io/optimized/_resources/js/vendor/responsive-tabs.js	JS (per function)	5 626	4 069 72.3 %	
https://gabrielashirley.github.lo/optimized/_resources/css/vendor/slick-theme.css	CSS	3 135	3 135100.0 %	
https://gabrielashirley.github.lo/optimized/_resources/css/vendor/slick.css	CSS	1 729	1 729100.0 %	
https://snap.licdn.com/ii.lms-analytics/insight.min.js	JS (per function)	3 578		
https://gabrielashirley.github.io/optimized/_resources/js/vendor/jquery.matchHeight.js	JS (per function)	11 703	706 6.0 %	
https://gabrielashirley.github.lo/optimized/	CSS+JS (per fu	2 186	380 17.4 %	
https://static.tagboard.com/embed/assets/is/embed.is	JS (per function)	5 858	144 2.5 %	
https://www.voutube.com/firame_api	JS (per function)	859	132 15.4 %	
https://w /a-message.isono?caliback=showEmergencyMessage& =1589843633118	JS (per function)	455	0 0.0 %	
https://helio.mvfonts.net/count/31ba49	CSS			
https://gabrielashirley.github.io/optimized/common/_emergency-broadcast/message.js	JS (per function)	2 228		
709 kB of 4.0 MB (18%) used so far. 3.3 MB unused.				

- a) CSS resources:
- _resources/css/vendor/brix_sans.css

This is a font declaration. It can be deferred later in favor of similar fallback fonts and reducing 2.1 MB load. Although it seems that this file is the biggest render-blocking of this page and has 100% unused bytes from the Coverage report, we cannot guarantee that this file is totally useless for the entire site unless we check coverage on every single page, while executing JavaScript, and under any possible combinations of state and media queries. And apparently, I found that some of the font-faces are used on some *button* elements on the homepage. Therefore, I chose to defer.

• _resources/css/custom-img.css

This file contains background-images that are used throughout the bottom sections of

the page. I chose to defer since it will not affect the first impression considering the limitation of viewports.

• _resources/css/custom.css

This file contains a lot of styles that are not used in the homepage so it is recommended to either remove or defer this resource. I chose to defer because it might be used in other pages, so the user can cache this resource for later-use.

resources/css/vendor/font-awesome.css

This is a font awesome integration usually for icons, and the only usage I found is for the left and right arrow on the carousel. I chose to defer this since the icon usage in the home page is minimal.

• _resources/css/vendor/slick.css resources/css/vendor/slick-theme.css

This comes from a slider vendor "Slick". I chose to defer this since it is only used in the "Academics" section in the middle of the page.

&

- b) JavaScript resources:
- https://cdnjs.cloudflare.com/ajax/libs/gsap/2.0.2/TweenMax.min.
 js & _resources/js/look-deeper.js

This is used for graphic animation in the "Look Deeper" section, although it is barely noticable in my opinion. I chose to defer, because deferring this would only make the animation load later after the initial render has occurred.

After taken the step I re-ran both Lighthouse and WebPageTest, the result is way better, the start render time improved from 3.2s to 1.2s:

						Web	Vitals	Do	cument Con	nplete		Fully Lo	baded	
	First Byte	Start Render	<u>First</u> <u>Contentful</u> <u>Paint</u>	<u>Speed</u> Index	<u>Last</u> <u>Painted</u> <u>Hero</u>	<u>Largest</u> <u>Contentful</u> <u>Paint</u>	<u>Cumulative</u> <u>Layout</u> <u>Shift</u>	Time	Requests	Bytes In	Time	Requests	Bytes In	Cost
First View (Run 1)	0.171s	1.200s	1.044s	2.898s	8.000s	1.044s	0.047	8.317s	104	4,303 KB	10.451s	139	5,071 KB	\$\$\$\$\$

							Web Vitals		Doc	ument Com	plete		Fully Loa	ded	
	First Byte	Start Render	<u>First</u> Contentful Paint	<u>Speed</u> Index	Last Painted Hero	<u>Largest</u> <u>Contentful</u> <u>Paint</u>	Cumulative Layout Shift	<u>Total</u> <u>Blocking</u> <u>Time</u>	Time	Requests	Bytes In	Time	Requests	Bytes In	Cost
First View (<u>Run 3</u>)	0.144s	0.200s	0.214s	2.646s	7.500s	0.947s	1.056	0.200s	8.249s	121	3,974 KB	10.350s	138	5,007 KB	<u>\$\$\$\$\$</u>

Performance Results (Median Run - SpeedIndex)

Waterfall View

		Start Re	ender	RUM F	irst Pair	t DC	M Intera	active	DO	M Co	ntent Loa	aded	0	n Load	Docu	ment C	Comple	te				
wait	dns	connect	S	51	htm]		js		CSS		imag	e	fla	ash	for	it	vi	deo	ot	her	JS E	Execution
															_						-	
Step 1				0.5	1.0	1.5	2.0 2	2.5	3.0 3.	5 4	.0 4.	55	.0 5.	.5 6.0	6.5	7.0	7.5	8.0	8.5	9.0	9.5	10.0
A 1 dabaiola	obinlo ub	io - impo	ound /	2/	0 00							1										
A 2 debrieles	shirleup	- briv can	oveu/	1							4	177 #	100 C									
A 3 dahriela	shirle _	custom-im	d ree		577	me					4	1// 0	10									
A d dahriela	chirle h	io - custo	m cee		605	me																
A 5. dabriela	shirle f	ont-auesom	e.css		604	ms																
A 6. dabriela	shirleh	in - slic	k.css		605	ms																
A 7 dahriela	shirle -	slick-them	e css		606	ms																
A 8. gabriela	shirleh.	in - stule	8.055		654	ms																
A 9. gabriela	shirlei	o – animat	e.css		654	ms																
A10. gabriela:	shirle b	ootstrap.m	in.is	and some state		387 0	s															
A11. gabriela:	shirleio	- slick.m	in.is			388 0	is is															
≜12. gabriela	shirlees	ponsive-ta	bs.is			389 p	ns l															
A13. gabriela:	shirleru	.easings.m	in.is				112	9 ms														
▲14. gabriela:	shirleio	- muscrip	ts.is				113	0 ms														
≜15. gabriela:	shirleb.	io - messa	ge.is				113	1 ms														
≜16. gabriela:	shirlery	.matchHeig	ht.js									38	06 ms									
▲17. gabriela:	shirleo	- logo_UCS	D.png				112	7 ms				1										
☐18. gabriela:	shirle	icon_searc	:h.png				112	8 ms				Т										
≜19. gabriela:	shirle	icon_arro	w.png									38	06 ms									
👌 20. gabriela:	shirleov	-2_homepag	e.jpg										4063	ms								
👌 21. gabriela:	shirle s	hrinkingic	e.jpg										4213	3 ms								
🛱 22. gabriela:	shirleou	nd_homepag	e.jpg											4568 m	s							
🛱 23. gabriela:	shirlei	o – icon_f	b.png											4568 m	s							
🛱 24. gabriela:	shirlei	o – icon_i	g.png											4595 m	s							
🛱 25. gabriela:	shirle	icon_linke	d.png											4596 🛛	IS							
🛱 26. gabriela:	shirlei	o – icon_t	w.png											4596 🛛	IS							
🛱 27. gabriela:	shirle	icon_chec	k.png											4596 m	IS							
🛱 28. gabriela:	shirle	look-deep	er.js											4597 🛛	IS							



2) Removed unused CSS and resource minification

I noticed that most of the JS and CSS files used are not minified yet and full of unused CSS. First, I did some byte shaving by changing *if else* on to ternary function, removing comments and dead codes, changing *function ()* to *arrow* => on *message.js* and *myscripts.js*.

I removed double declarations on background images on *custom-img.css*, using both *background* and *background-image* to refer to the same file and the same class selector. Moreover, there are some double even triple declarations on certain selectors as well such as *.attend-module* and *.attend-cta*.

Next, I found that @font-face in *font-awesome.css* has missing source files (it only has *woff2*) so I used only the *woff2* format and removed the other formats since *woff2* already covers 95.23% users of modern browsers with upgraded compression. The same thing goes for *glyphicons-halflings-regular.woff2* under *bootstrap/fonts*, so I only kept that one as the source file on *styles.css*. I also removed unused font awesome icons which cut around 2,000 lines of codes, since the homepage seems to only use left and right arrow icons.

Out of curiosity, I tried another hack on our biggest CSS problem *brix_sans.css*. I modified it to extract fonts to separate files under _resources/fonts/brix_sans. For controlling font performance, I applied *font-display: swap* as it has no block period and infinite swap period ensure text remains visible webfont to during load (https://developers.google.com/web/updates/2016/02/font-display). I also removed the 2 font-face declarations that are already commented out, and moved the @import url ("//hello.myfonts.net/count/31ba49"); to the bottom since this unnecessary import of an analytics URL seems to cause the block. When I see the Network tab and disable cache, this import takes a long time to load, perhaps the *myfonts* server is the problem.

Since I feel like I have done much on the manual deletion, I minified JS and CSS files using minifier tool online <u>https://javascript-minifier.com/</u> and <u>https://cssminifier.com/</u> for: *custom-img.css, custom.css, brix_sans.css, animate.css, font-awesome.css, slick-theme.css, slick.css, look-deeper.js, myscripts.js, jquery-matchHeight.js, responsive-tabs.js,* and *message.js.* This step shows a bit of improvement on the performance score:



Performance Results (Median Run - SpeedIndex)

						Web	Vitals	Doo	cument Con	nplete		Fully L	oaded	
	First Byte	Start Render	<u>First</u> Contentful <u>Paint</u>	<u>Speed</u> Index	Last Painted Hero	<u>Largest</u> <u>Contentful</u> <u>Paint</u>	<u>Cumulative</u> <u>Layout</u> <u>Shift</u>	Time	Requests	Bytes In	Time	Requests	Bytes In	Cost
First View (Run 3)	0.131s	0.800s	0.646s	1.368s	2.200s	2.191s	0.01	4.902s	117	2,212 KB	7.028s	147	3,173 KB	\$\$\$\$\$

Additionally, based on the networking data, the currently optimized site has 8.9 MB transferred and 14.0 MB resources whereas the originally mirrored site has 17.5 MB transferred and 23.8 MB resources. This shows great progress in all our minification attempts.

3) Image compressions and proper sizing

It seems that some of the images have been compressed well enough, there are no excessively large images > 1MB and most of the small web graphics are < 300KB. However, there are still some potentially compressed images and improper sizes. So I compressed the JPEG files and made it as progressive JPEG using http://optimizilla.com/. I changed one UCSD logo to download it into the local directory instead of fetching http://www.ucsd.edu/ resources/img/logo UCSD white.png.

Modified images:

- Compressed:
 - i. _resources/img/bg_attend.jpg Result: from 587.9 KB to 253.3 KB
 - ii. _resources/img/bg_map.jpg Result: from 94.0 KB to 45.6 KB
 - iii. _images/homepage/bg_studentlife_alt2.jpg
 Result: from 244.3 KB to 62.1 KB
 - iv. _images/The-Playground_homepage.jpg Result: from 141 KB to 66 KB
 - v. _images/shrinkingice.jpg Result: from 82 KB to 8.7 KB
 - vi. _images/homepage/u-care-ucsd-1400x439.jpg Result: from 17 KB to 14.1 KB
 - vii. _images/home-financial-aid.jpg
 Result: from 54 KB to 47.7 KB
- Resized:
 - i. _images/sars-cov-2_homepage.jpg

Status: resized to 450px wide as shown on https://ucsdnews.ucsd.edu/pressrelease/introducing-the-uc-san-diego-return-to-le arn-program

Result: from 104 KB to 38 KB

- ii. _images/The-Playground_homepage.jpg
 Status: resized to 265px wide since as shown on home page
 Result: from 66 KB to 25.4 KB
- iii. _images/stats-number-one-public-service.jpg Status: resized to 310px high as shown on home page through CSS rules Result: from 45 KB to 31 KB
- iv. _images/stats-second-quality-education.jpg
 Status: resized to 310px high as shown on home page through CSS rules
 Result: from 58 KB to 37 KB

V. _images/stats-top-ten.jpg

Status: resized to 310px high as shown on home page through CSS rules Result: from 71 KB to 41 KB

The improvement seems pretty good, we 100% compressed all images and use progressive JPEGs.

50 67 (Performance Accessibility Pr	79 89 Best SEO Actices Web App
- 0-49 -	50-89 🗕 90-100
Perfo	50 ormance
Metrics	
First Contentful Paint 2.3 s	First Meaningful Paint 3.9 s
Speed Index 4.6 s	First CPU Idle 5.5 s
▲ Time to Interactive 19.8 :	Max Potential First Input Delay 1,080 ms

Compress Images: 100/100 Learn More

679.5 KB total in images, target size = 679.5 KB - potential savings = 0.0 KB

Use Progressive JPEGs: 100/100 Learn More

646.1 KB of a possible 646.1 KB (100%) were from progressive JPEG images

							Web Vitals		Doc	ument Com	plete		Fully Loa	aded	
	First Byte	Start Render	<u>First</u> <u>Contentful</u> <u>Paint</u>	<u>Speed</u> Index	Last Painted Hero	<u>Largest</u> <u>Contentful</u> <u>Paint</u>	Cumulative Layout Shift	<u>Total</u> <u>Blocking</u> <u>Time</u>	Time	Requests	Bytes In	Time	Requests	Bytes In	Cost
First View (<u>Run 1</u>)	0.131s	0.700s	0.600s	1.293s	1.700s	0.600s	0.047	≥ 0.419s	5.034s	121	2,170 KB	7.346s	146	3,281 KB	<u>\$\$\$\$\$</u>

Performance Results (Median Run - SpeedIndex)

4) HTML Formatting (removed unused line breaks), Extracted SVG, and Utilized CDN for several vendor resources

My first impression looking at the index.html file is the messy, excessively long code with too many unused line breaks. So I formatted the document using VSCode built-in HTML Language Feature formatter, and removed all the unused line breaks. Then, I extracted the 2 SVG elements to *hero-big.svg* and *hero-small.svg* under _*resources*:

<object data="_resources/hero-big.svg" type="image/svg+xml" class="hide-for-small-only"></object>
<object data="_resources/hero-small.svg" type="image/svg+xml" class="show-for-small-only"></object>

Next, I utilized CDN usage for several vendor resources: slick.min.css, slick-theme.min.css, jquery.matchHeight-min.js, and slick.js.

<script src="_resources/js/vendor/jquery.matchHeight.js" > <script crossorigin="anonymous" src="https://cdnjs.cloudflare.com/ajax/libs/jquery.matchHeight/0.7.2/jquery.matchHeight-min.js"></script>
<script src="_resources/js/vendor/slick.min.js" type="text/javascript" > <script <i="">src</i>="<u>https://cdnjs.cloudflare.com/ajax/libs/slick-carousel/1.6.0/slick.js</u>" <i>crossorigin</i>="anonymous"></script>
<pre><!-- Slick Slider for custom carousel--> <!-- <link href="_resources/css/vendor/slick.css" rel="stylesheet" type="text/css"-->> <link "="" as="style" href="https://cdnjs.cloudflare.com/ajax/libs/slick-carousel/1.6.0/slick.min.css" onload="this.onload=null;this.rel='stylesheet"/> <!-- <link href="nettop://cdnjs.cloudflare.com/ajax/libs/slick-carousel/1.6.0/slick.min.css" as="style" onload="this.onload=null;this.rel='stylesheet""--> <!-- <link href="nettop://cdnjs.cloudflare.com/ajax/libs/slick-carousel/1.6.0/slick.min.css" as="style" onload="this.onload=null;this.rel='stylesheet""--> <!-- <link href="nettop://cdnjs.cloudflare.com/ajax/libs/slick-carousel/1.6.0/slick-theme.min.css" as="style" onload="this.onload=null;this.rel='stylesheet""--> <!-- <link href="nettop://cdnjs.cloudflare.com/ajax/libs/slick-carousel/1.6.0/slick-theme.min.css" as="style" onload="this.onload=null;this.rel='stylesheet"" </ </ </ </ </ </ </--></pre>

Performance Results (Median Run - SpeedIndex)

							Web Vitals		Doc	ument Com	plete		Fully Lo	aded	
	First Byte	Start Render	<u>First</u> <u>Contentful</u> <u>Paint</u>	<u>Speed</u> Index	Last Painted Hero	<u>Largest</u> <u>Contentful</u> <u>Paint</u>	Cumulative Layout Shift	<u>Total</u> <u>Blocking</u> <u>Time</u>	Time	Requests	Bytes In	Time	Requests	Bytes In	Cost
First View (<u>Run 2</u>)	0.165s	0.600s	0.453s	1.111s	1.300s	0.589s	0.022	≥ 0.171s	5.000s	139	2,141 KB	7.118s	148	3,264 KB	<u>\$\$\$\$\$</u>

This step apparently does not show any significant improvement from before.

5) Extract Critical CSS

Since our current goal is to improve the web performance, I chose to sacrifice the code readability and purify the CSS to only one main file *purified.css*, resulting in less HTTP requests and staying DRY (Don't Repeat Yourself). I used <u>https://github.com/purifycss/purifycss</u> and it reduced all the CSS files by ~25.5%. This step significantly raises the performance score from 50 to 70.



Performance Results (Median Run - SpeedIndex)

							Web Vitals		Doc	ument Com	plete		Fully Lo	aded	
	First Byte	Start Render	<u>First</u> <u>Contentful</u> <u>Paint</u>	<u>Speed</u> Index	Last Painted Hero	Largest Contentful Paint	Cumulative Layout Shift	<u>Total</u> <u>Blocking</u> <u>Time</u>	Time	Requests	Bytes In	Time	Requests	Bytes In	Cost
First View (Run 2)	0.134s	0.500s	0.412s	1.041s	1.300s	0.523s	0.023	≥ 0.223s	4.880s	119	2,124 KB	7.250s	141	3,264 KB	\$\$\$\$\$

Conclusion

Up until this point, I have tried to optimize the mirror of https://ucsd.edu/ from 3 to 70 https://gabrielashirley.github.io/improved/. Reading the baseline report from Lighthouse and WebPageTest, I quickly noticed how bad the render-blocking CSS affects the performance, especially the start render time metric. So I manipulated the JS and CSS load using *defer* and *rel=preload* which gave score 31. I then removed unused CSS manually and minified all JS and CSS files, I modified the huge *brix_sans.css* too. This step cut ~42% resources and ~20% transferred according to the Network tab. Next, I optimized images by compressing and resizing. This shows quite an increase to 50. Then, I removed unused line breaks on HTML, extracted 2 SVGs, and utilized CDN for some vendor resources. And last but not least, I extracted critical CSS into one file which reduced the CSS file size by ~25.5%. This step apparently solved another performance bug from this site, adding 20 more points to the performance score. So in the meantime, I got 70, but hopefully I could work on some more improvements in the future.