Optimizing JShelter performance

Martin Zmitko

Supervisor: Ing. Radek Hranický, Ph.D.

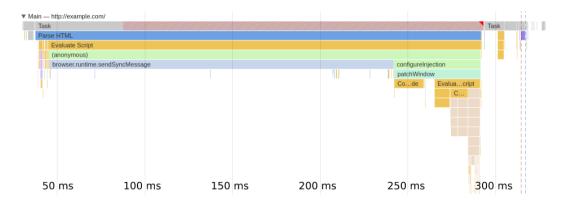


May 23, 2023

Chrome – Regular Injection Performance



- On almost every load
- 250 ms performance hit
- Slow SyncMessage 180 ms on every load



SyncMessage Performance



- Large payload size 700 kB
- Necessary serialization and deserialization
- 30 ms spent by background script handler
- Slow internal browser processing
- Linear execution time increase

· //		200 ms		400 m	15		6	300 ms	800 ms	100	0 ms	1200 ms	1400 ms	1600 ms		1800		2000 ms	2200	ms		0 ms	2 11/11
rms F	rames	2232 ms s	223	4 ms	2	236 п	ns	2238 ms	2240 m	is 2242 π	15 2244 m	s 2246 m	is 2248 ms	2250 ms	225	2 ms	2254 ms	2256 ms	22	58 ms	2260	15	2262 ms
ΨN	fain —	- chrome-e	xtension://	fpomppg	ndpe)	ekpfft	gedgi	mnagjijgli/_ge	enerated_backg	pround_page.htm	h												
	Task										Task		Task	Task									
		tion Call					Run I	Microtasks				. F.,	I Major GC	Function Call									
		Major GC		suk				(anonymou						subEback			rentCallback				EventCallba	.k	
				tad>	t.,	.>		storeAsyno						targeted>		target	<computed></computed>			up	date	upe	
				Apply	Α.	у				M M				Apply		Apply							
				onst	0.	t								onBefues	t i	onBef	oreRequest						
				nors		s								asyncRet		async	Ret						
				coer	C.,	r								ret		ret							
				gen																			
				(as)	(.)																	
				rese		.е							1										
				fpde																			- F
					ч																		

Chrome – Early Injection Performance

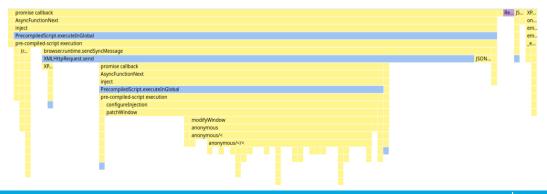


- Uncommon
- 80 ms performance hit
- 10 ms for evaluation, 20 ms added to total time

45 ms	s 95 ms	s 145 ms	195 ms	245 ms	295 ms	345 ms	395 m	s 445	ms 49	5 ms 5	45 ms 5	595 ms	645 ins (595 ns	745 ms
<u>Un III.</u>											- And			11.	
505 ms ► Síť	515 ms	525 ms	535 ms	545 ms	555 ms	565 ms	575 ms	585 ms	595 ms	605 ms	615 ms	625 ms	635 ms	645 ms	655
▼ Hlavní —	https://exam	ple.com/													
Úla Úlo	oha				//////									8	
An	alýza HTML														
		Vypt		V	Vyhodnotit sk										
		Zkt			Zkompilov	/at kód			notit skript						
		Zkd						(anonyi	mní)						
								doConf	figuration						
								configu	reInjection						
								patchW	/indow				111		
								insertBe	efore						
								Zkomp	iskript Zk	ompilovat k	ód				
								Zkomp	it kód						

Firefox Injection Performance

- Always the same
- 80 ms performance hit
- Injected as a content script patchWindow executes during SyncMessage handling
- Necessary to complete request, additional 20 ms





Wrapper Performance

- Small performance hit for all
- Most couldn't be further optimized
- Large performance hit on farbling
- Inefficient iteration

```
0.2 ms let crc = new CRC16();
0.2 ms for (row of rowIterator()) {
0.5 ms
            crc.next(row);
        var thiscanvas prng = alea(domainHash, "CanvasFarbling", crc.crc);
        var data count = BigInt(BigInt(width) * 4n);
0.3 ms for (row of rowIterator()) {
25.1 ms
            for (let i = 0n; i < data count; i++) {</pre>
7.8 ms
                if ((i \% 4n) == 3n) {
                    // Do not modify alpha
                    continue:
                if (thiscanvas prng.get bits(1)) { // Modify data with probability of 0.5
 1.5 ms
                    // Possible improvements:
                    // Copy a neighbor pixel (possibly with modifications
                    // Make bigger canges than xoring with 1
90.3 ms
                    row[i] ^= 1:
```





- Decrease SyncMessage payload size don't send code
- Split configuration and code generation logic
- Move code generation to content scripts
- Generate code in document_start.js
- Wrapper definition evaluation adds 10 ms, code generation takes 15 ms
- Final SyncMessage payload size is 12 kB and executes under 10 ms
- Code size optimizations

WebAssembly optimized farbling

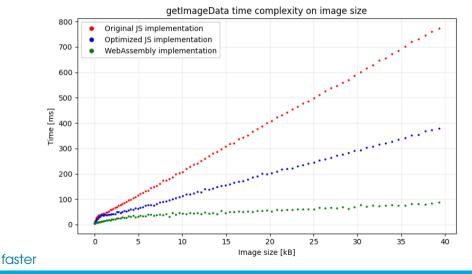


- Allows efficient data processing
- Subject to CSP on Chrome
- Modify CPS headers, adjustable level
- Inconsistent initialization
- JS implementation used as a fallback, optimized implementation must always provide same results
- Not subject to CSP on Firefox, possible to use WebAssembly only
- Reimplemented Canvas, WebGL and WebAudio farbling in AssemblyScript TypeScript syntax, for compiling into WebAssembly
- Differences between number types and operations
- Automatic build process
- Unit tests
- Known bug: floating point CRC provides different results

Optimized canvas farbling measurement



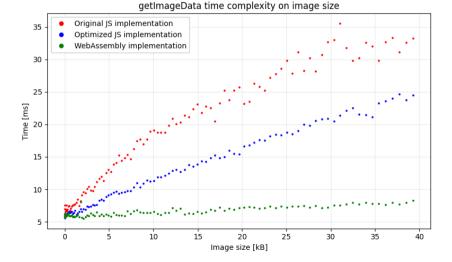
Measured on Chrome for square canvas with data in range 0.4-4000 kB, 5.3 times



Optimized canvas farbling measurement detail



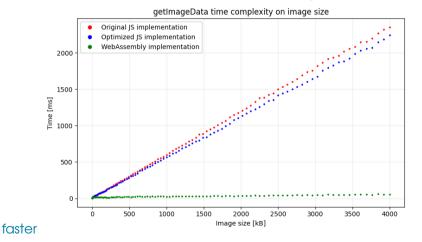
Measured on Chrome for square canvas with data in range 40 B to 40 kB



Optimized canvas farbling measurement



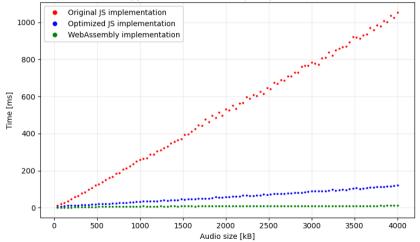
Measured on Firefox for square canvas with data in range 0.4-4000 kB, 53 times



Optimized audio farbling measurement



Measured on Chrome for audio in range 0.4-4000 kB

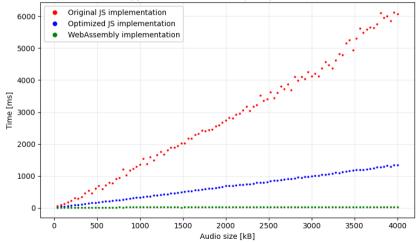


copyFromChannel time complexity on audio size

Optimized audio farbling measurement



Measured on Firefox for audio in range 0.4-4000 kB



copyFromChannel time complexity on audio size

Lighthouse loading analysis

T FIT

- Tool for measuring user percieved loading performance
- No similiar tools found
- Implemented own CLI tool for collecting performance data on set URLs with set extensions
- JSON output for analysis
- Measured on 50 top Tranco domains
- Performance of clean browser was 83.5, original JShelter 69.2 and optimized JShelter 78.5
- That is 13,5% increase from original version
- Original version decreased performance by 17,2 %, optimized version decreased performance just by 6,1 %