

Mega-fast or just super-fast? Performance differences of mainstream JavaScript frameworks for web applications

Andreas Nicklaus

17.10.2024

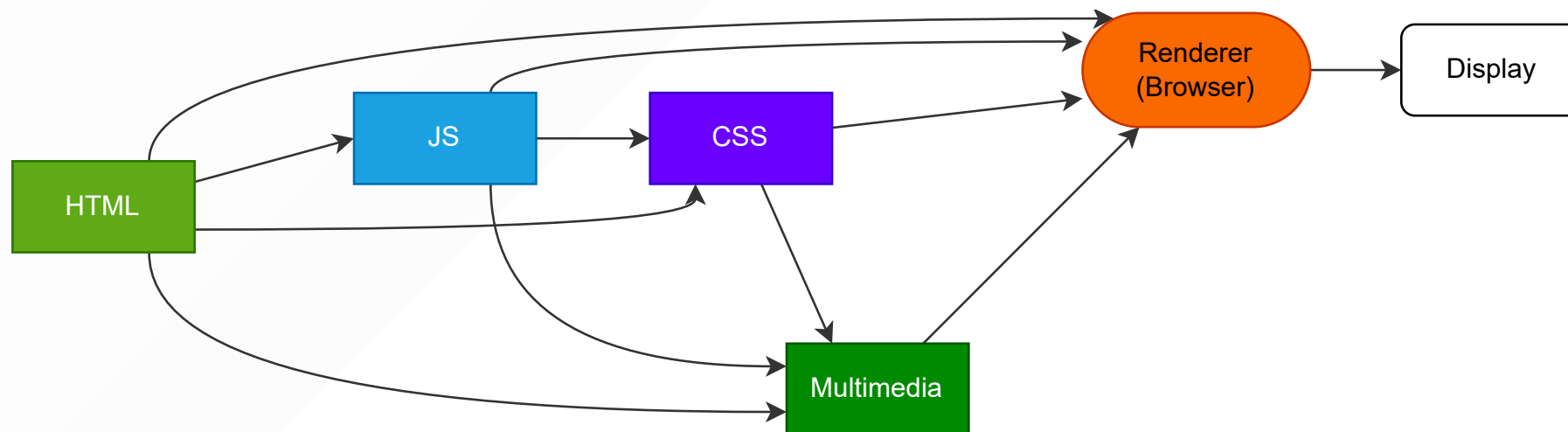


Agenda

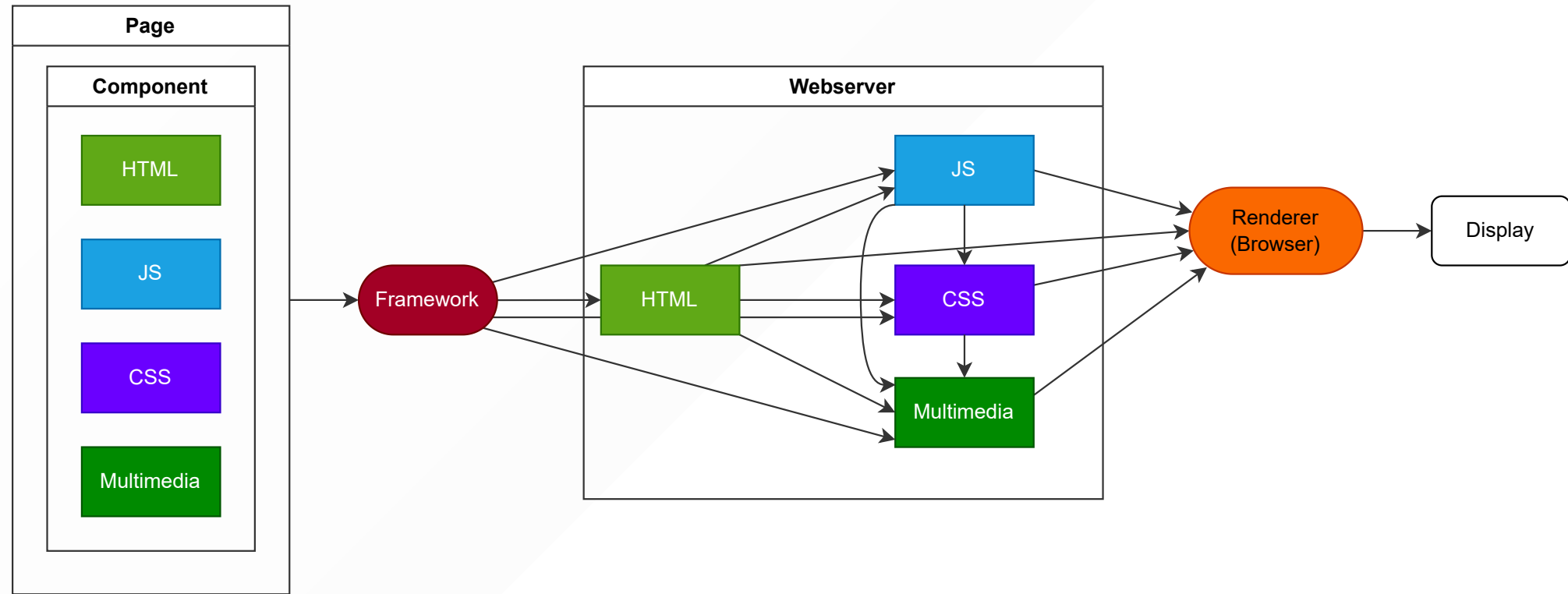
1. Themenübersicht
2. Fragestellung
3. Lösungsstrategie und -design
4. Ergebnisse
5. Lessons Learned

1. Themenübersicht

“ Websites are made up of files written in HTML and CSS that are rendered and displayed in web browsers. They can be static, with pre-defined content, or dynamic, changing automatically based on user input or other factors. [[1](#)]



Mega-fast or just super-fast? Performance differences of mainstream JavaScript frameworks for web applications



2. Fragestellung

“ Hat die Wahl des Frameworks einen für den Nutzer merklichen Einfluss auf die Render-Geschwindigkeit der Webseite? ”

2. Fragestellung

“ Hat die Wahl des Frameworks einen für den Nutzer merklichen Einfluss auf die Render-Geschwindigkeit der Webseite? ”

- Framework

2. Fragestellung

“ Hat die Wahl des Frameworks einen für den Nutzer merklichen Einfluss auf die Render-Geschwindigkeit der Webseite? ”

- Framework
- für den Nutzer merklich

2. Fragestellung

“ Hat die Wahl des Frameworks einen für den Nutzer merklichen Einfluss auf die Render-Geschwindigkeit der Webseite? ”

- Framework
- für den Nutzer merklich
- Render-Geschwindigkeit

3. Lösungsstrategie und -design

1. Frameworks
2. Anwendung: Seiten, Komponenten und Content
3. Hosting-Umgebung
4. Metriken & Untersuchungsgegenstand
5. Test-Tools
6. Browser

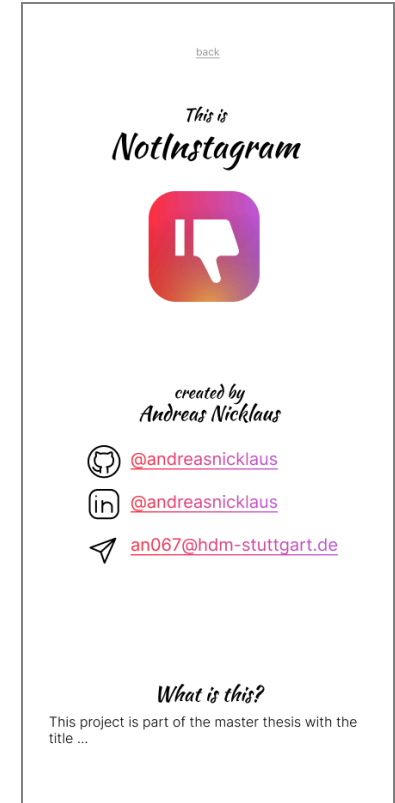
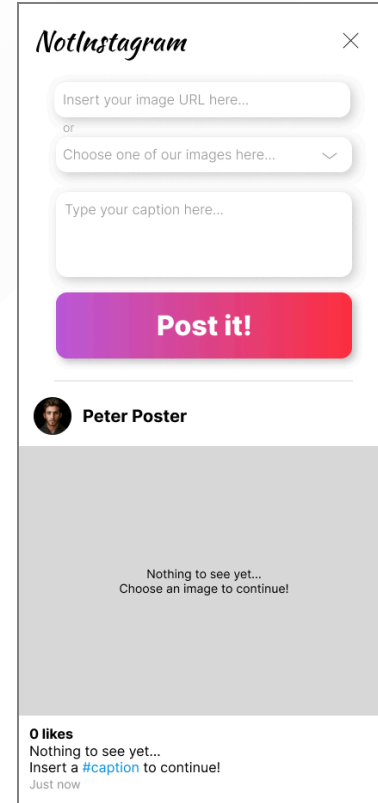
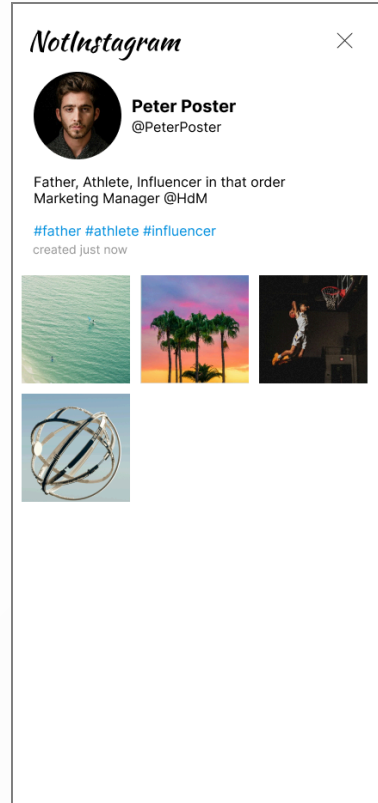
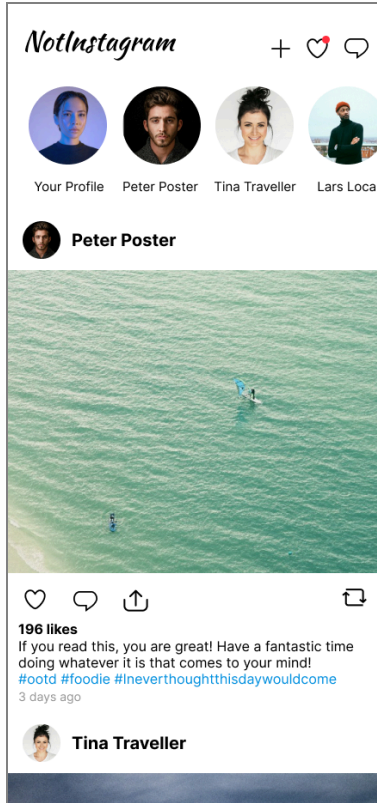
3.1 Frameworks

CSR	SSR
Angular	Astro
React	Next.js
Vue.js	Nuxt
Svelte	

Entscheidungskriterien: [\[2\]](#)

- Nutzungsquote
- Empfehlungsrate

3.2 Beispielanwendung



3.3 Hosting-Umgebung

Vercel [3]

- Network Delay
- Kostenloses Konto
- CI/CD Integration

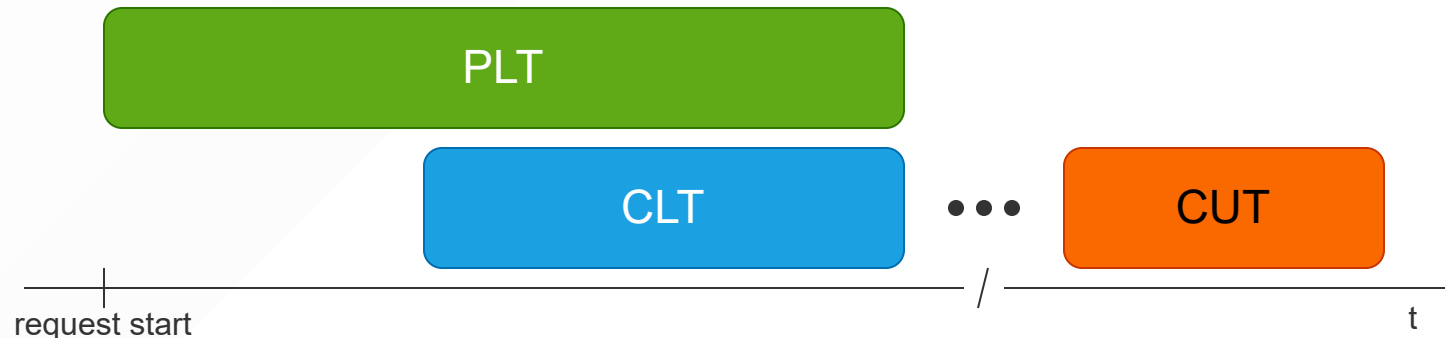
Localhost

- Reine Render-Geschwindigkeit
- `serve` oder Framework Preview
- Baseline ohne Netzwerkverzögerung

3.4 Metriken

3 Kategorien zur besseren Unterteilung:

- Page Load Time (PLT)
- Component Load Time (CLT)
- Component Update Time (CUT)



	PLT	CLT	CUT
Total Byte Weight	x		
Time To First Byte	x		
DomContentLoaded	x		
Last Visual Change	x		
Largest Contentful Paint	x		

	PLT	CLT	CUT
Time To Interactive	X	X	
Total Blocking Time	X	X	
LoadEventEnd	X	X	
Observed First Visual Change		X	
Observed Last Visual Change		X	
DOM Mutation Times		X	X

3.5 Test-Tools

Lighthouse CLI

- State of the Art für Web-Performance
- Umfangreiche Sammlung an Metriken
- Automatisierung von Tests
- Google Chrome

[4]

Playwright

- Tests für Content und Interaktionen
- Custom Tests
- Injektion von Skripts
▶ Black-Box-Testing
- Freie Browser-Wahl

[5]

Lighthouse	Playwright
Total Byte Weight (TBW)	DomContentLoaded
Time To First Byte (TTFB)	loadEventEnd
Time To Interactive (TTI)	Mutation Times
Total Blocking Time (TBT)	
Largest Contentful Paint (LCP)	
Observed First Visual Change (OFVC)	
Observed Last Visual Change (OLVC)	

3.6 Browser

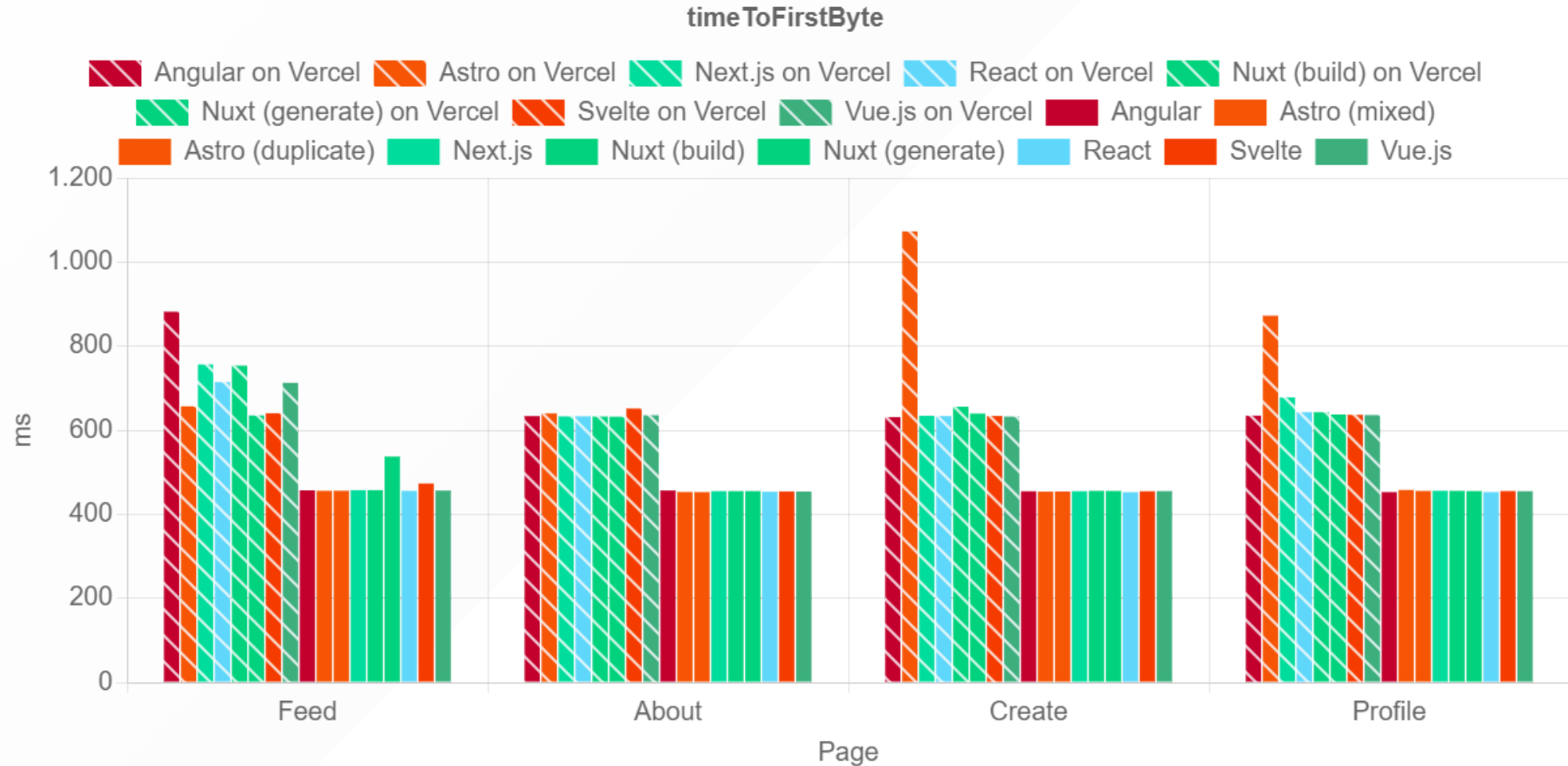
- Google Chrome
- Mobile Chrome
- Chromium
- Microsoft Edge
- Firefox
- Desktop Safari
- Mobile Safari

[6]

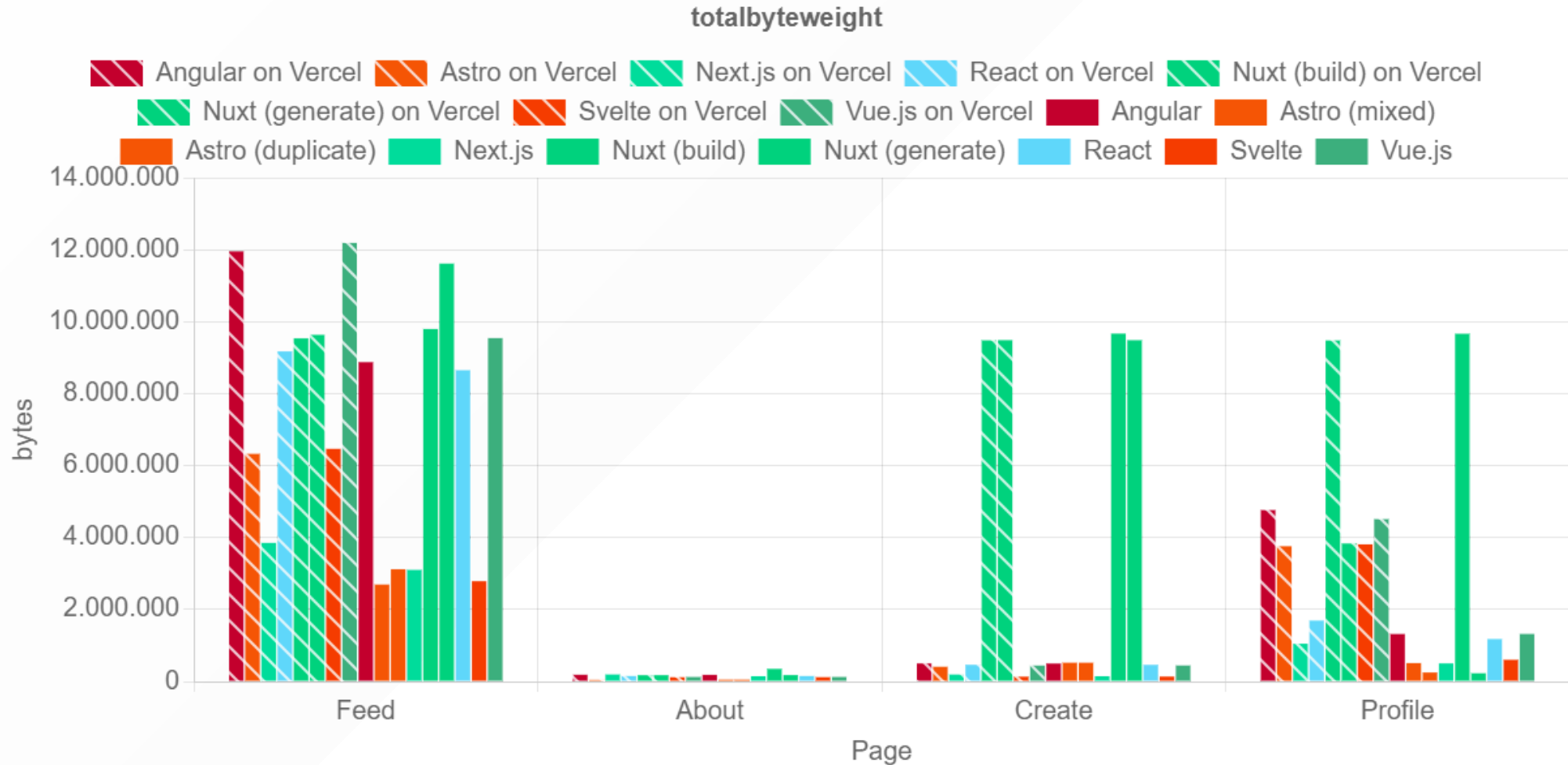
4. Ergebnisse

- Unschlüssig für PLT und CLT
 - Ungleich verteilte Stärken und Schwächen der Frameworks
 - Undeutliche Tendenzen bzgl. Client-Side vs. Server-Side Rendering
- Undeutlich für CUT
 - Zeiten und Zeitspannen der DOM Mutations
 - Durchschnittsrangking von Frameworks und Browsern möglich

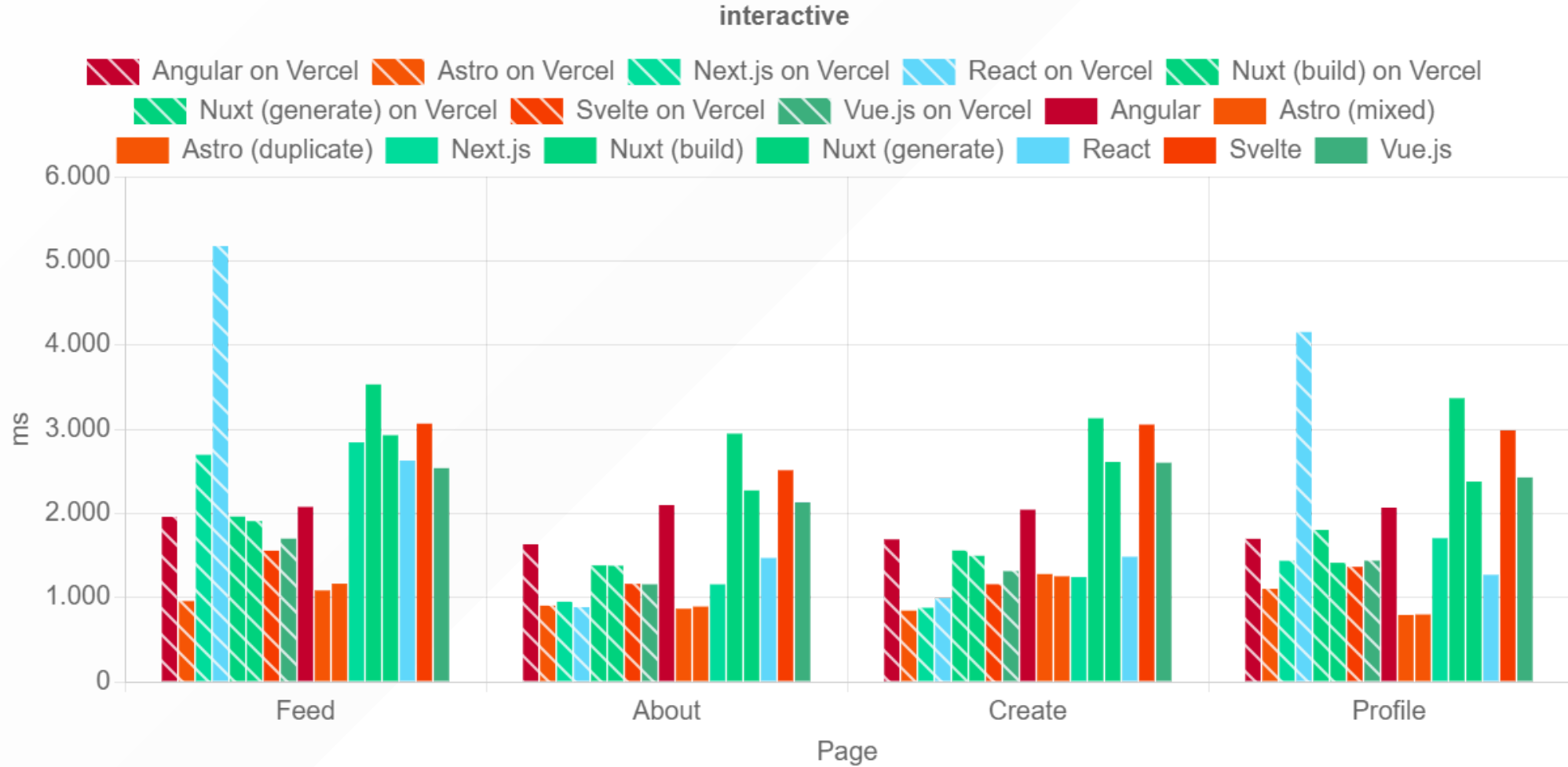
4.1 Page Load Time - TTFB



4.1 Page Load Time - TBW



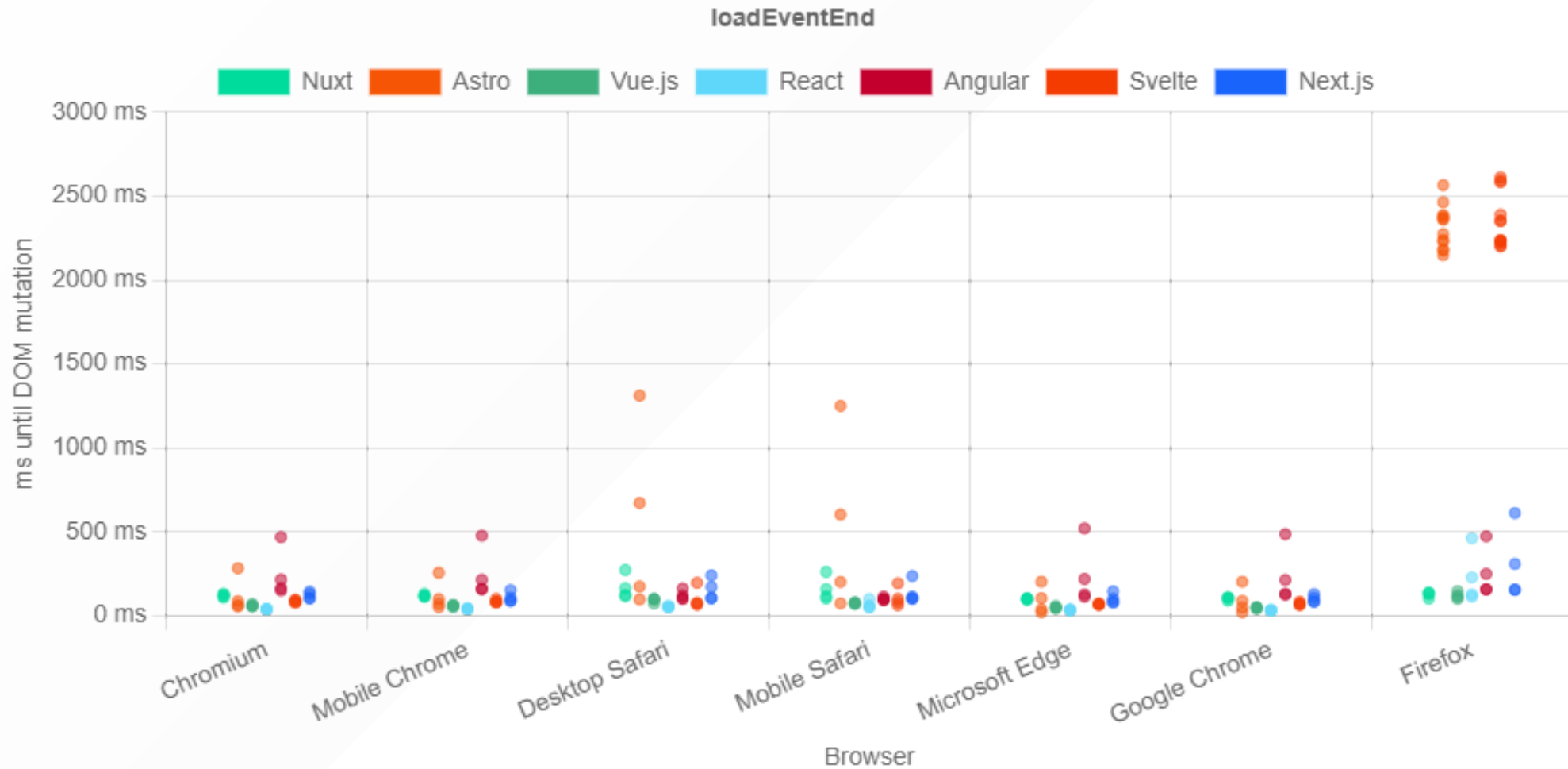
4.1 Page Load Time - TTI



4.1 Page Load Time - DomContentLoaded



4.2 Component Load Time - LoadEventEnd

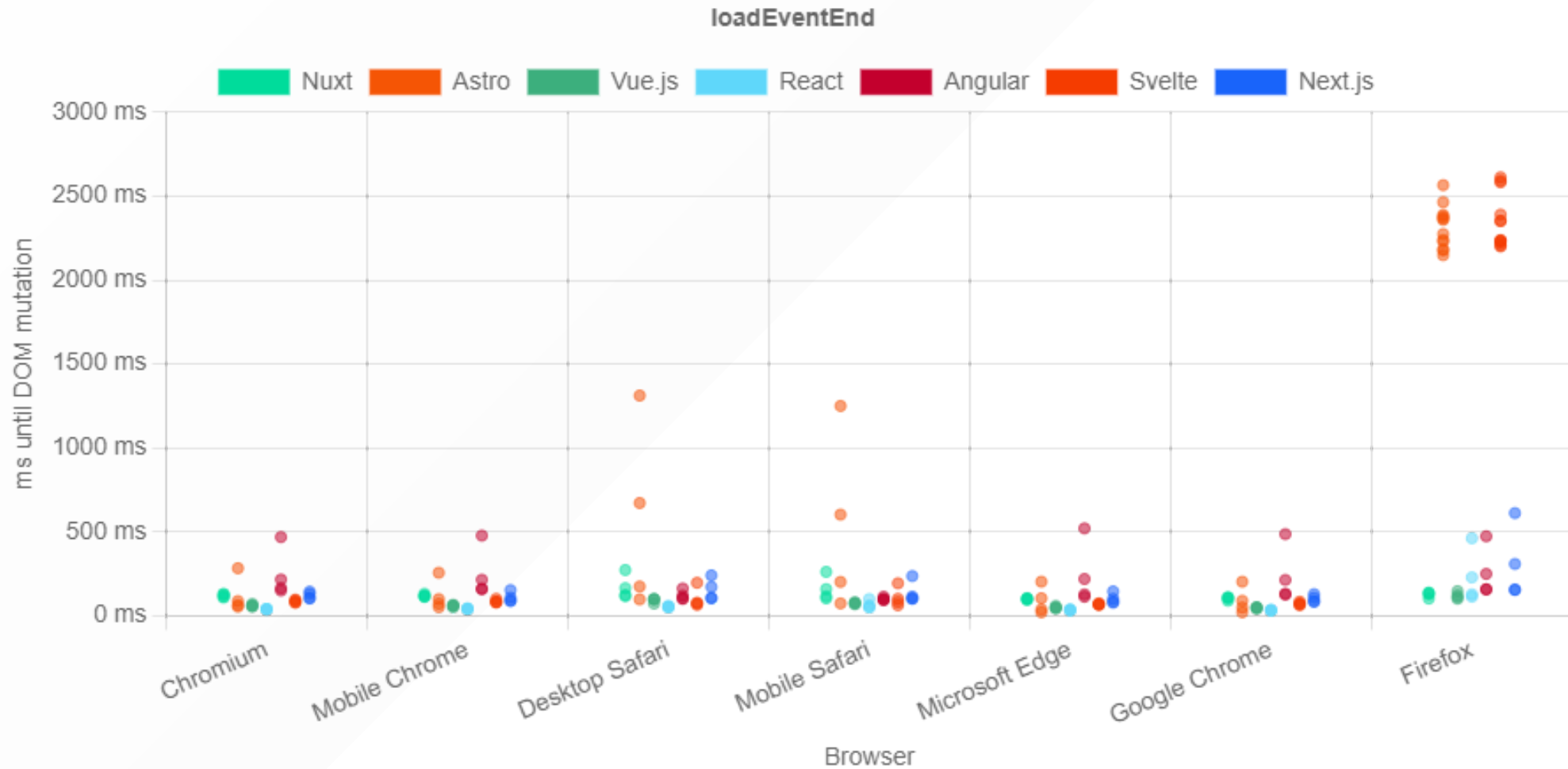


4.2 Component Load Time - balanced LoadEventEnd (1)

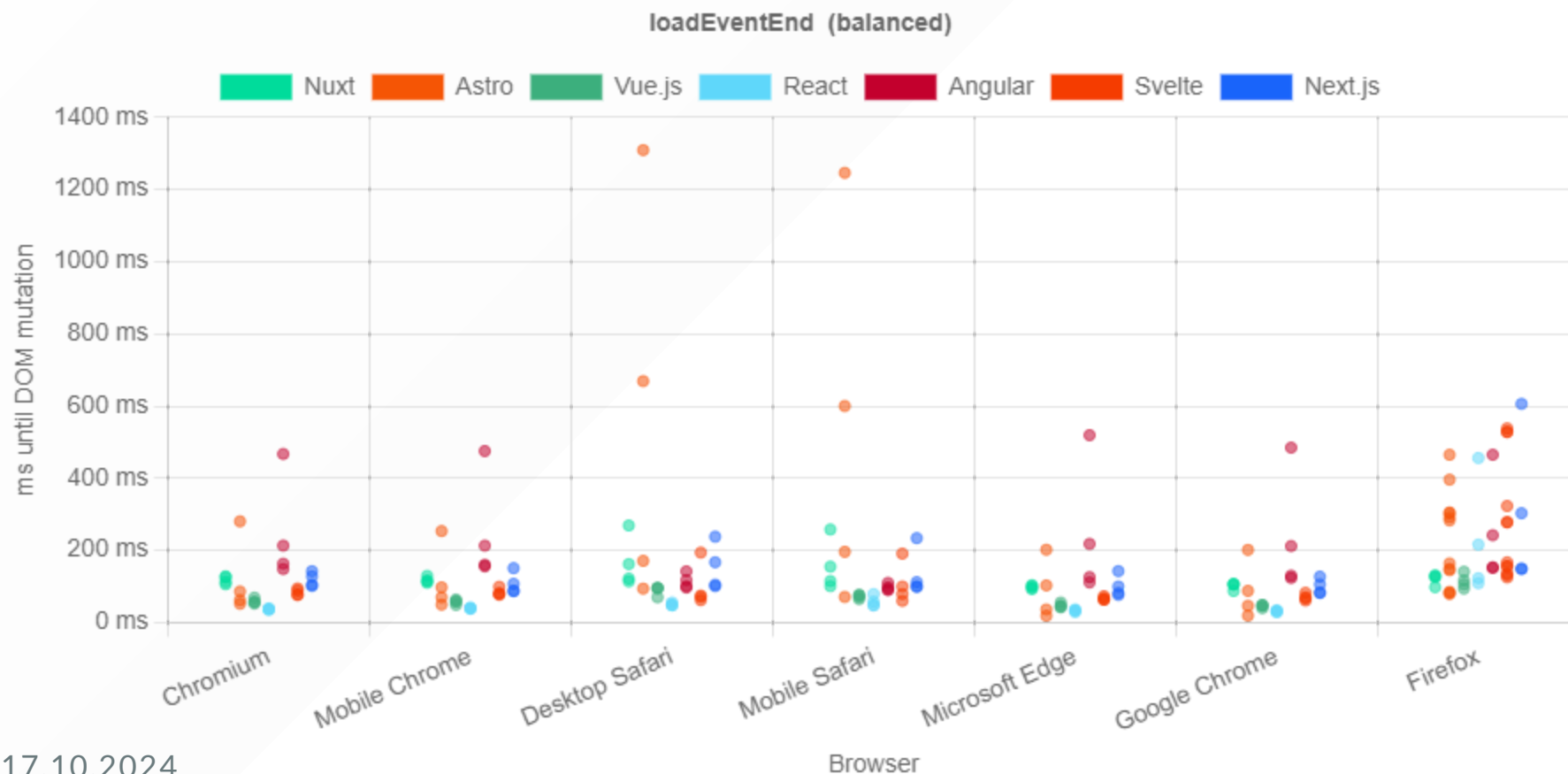
In Firefox werden Requests teilweise erst verspätet gemacht. Das balanced LoadEventEnd nur misst die Zeit nach dem Requeststart.

$$\mathit{loadEventEnd}_{\mathit{balanced}} = \mathit{loadEventEnd}_{\mathit{raw}} - \mathit{requestStart}$$

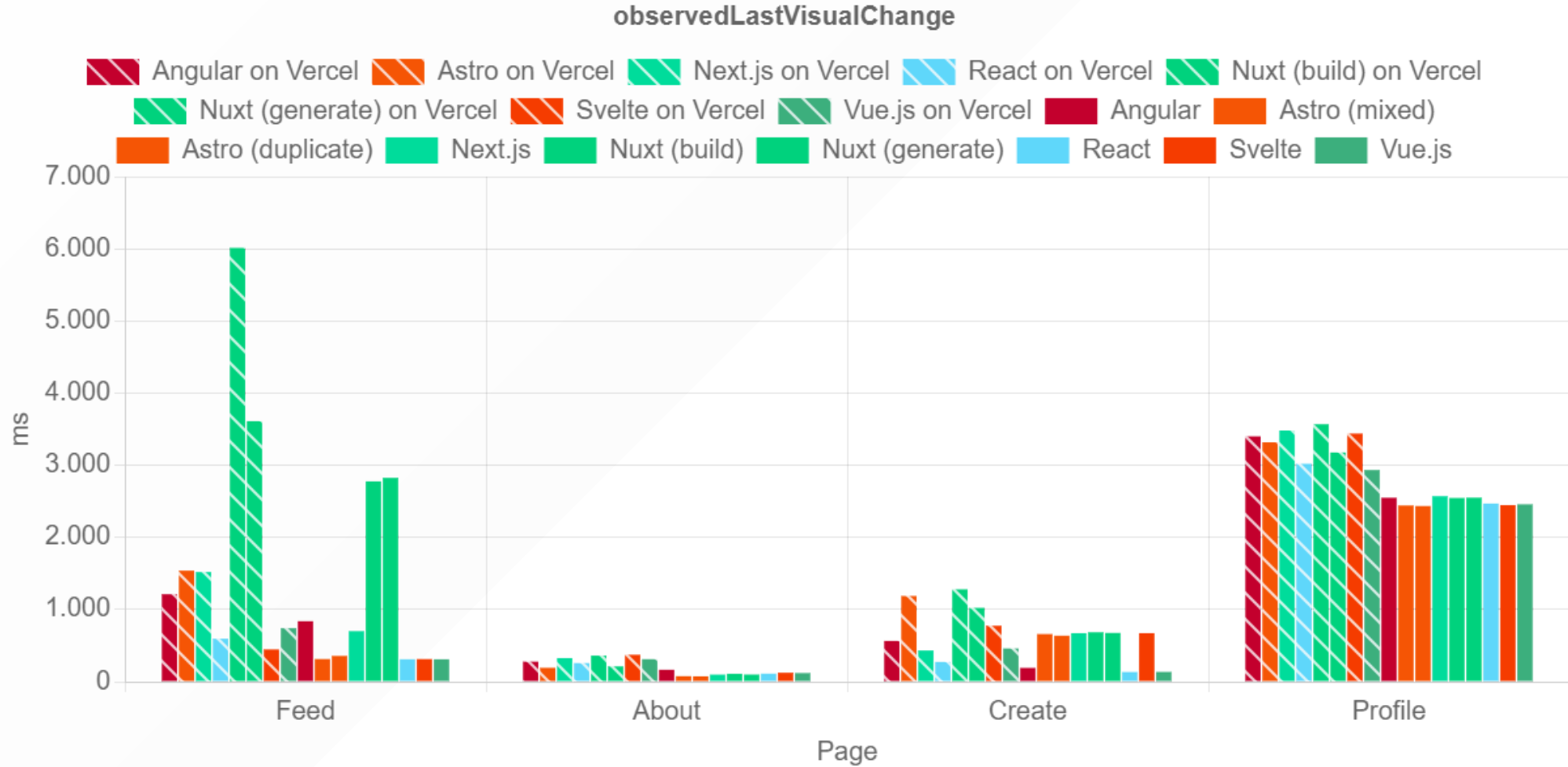
4.2 Component Load Time - LoadEventEnd



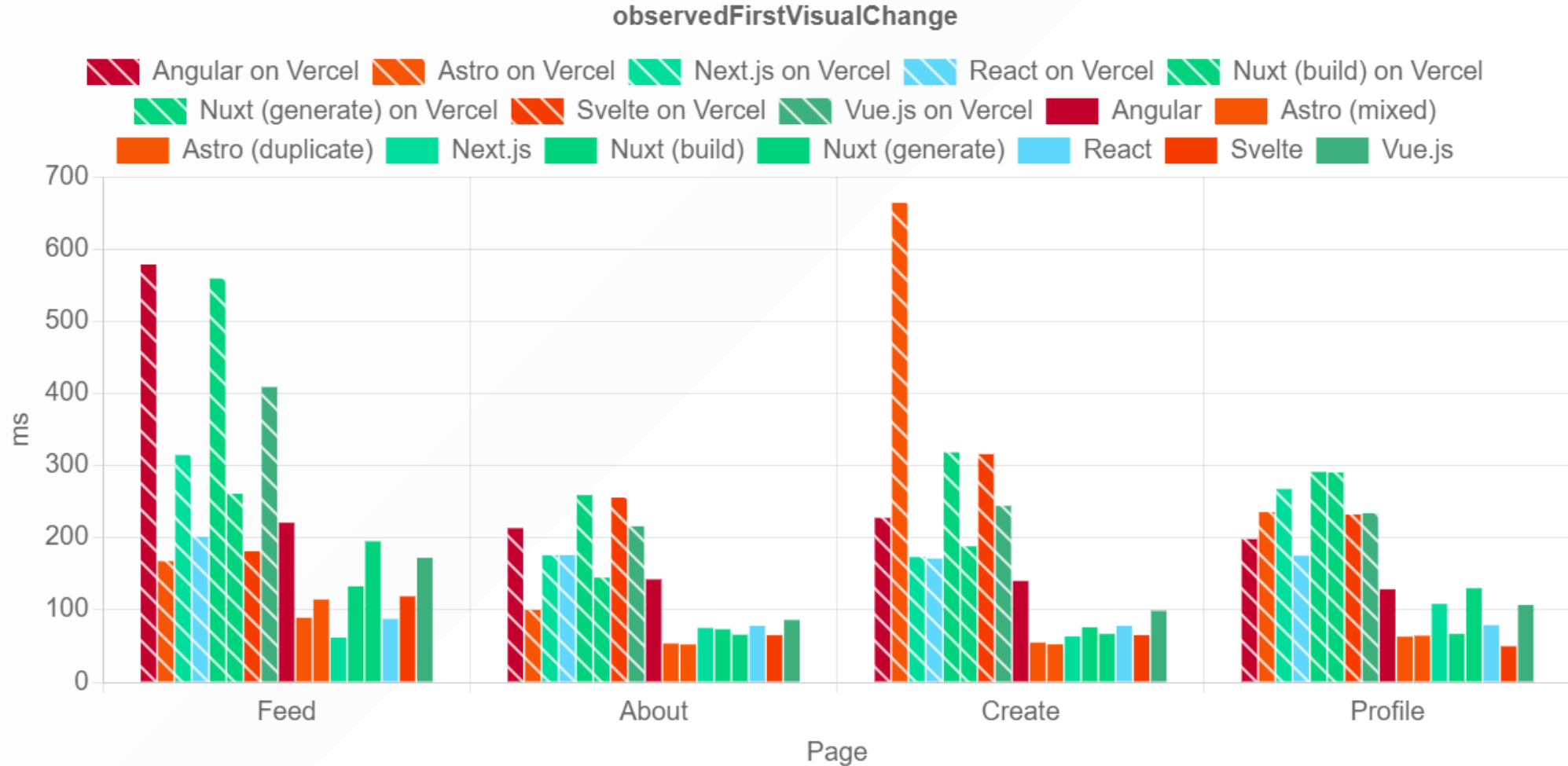
4.2 Component Load Time - balanced LoadEventEnd (2)



4.1 Page Load Time - OLVC



4.2 Component Load Time - OFVC

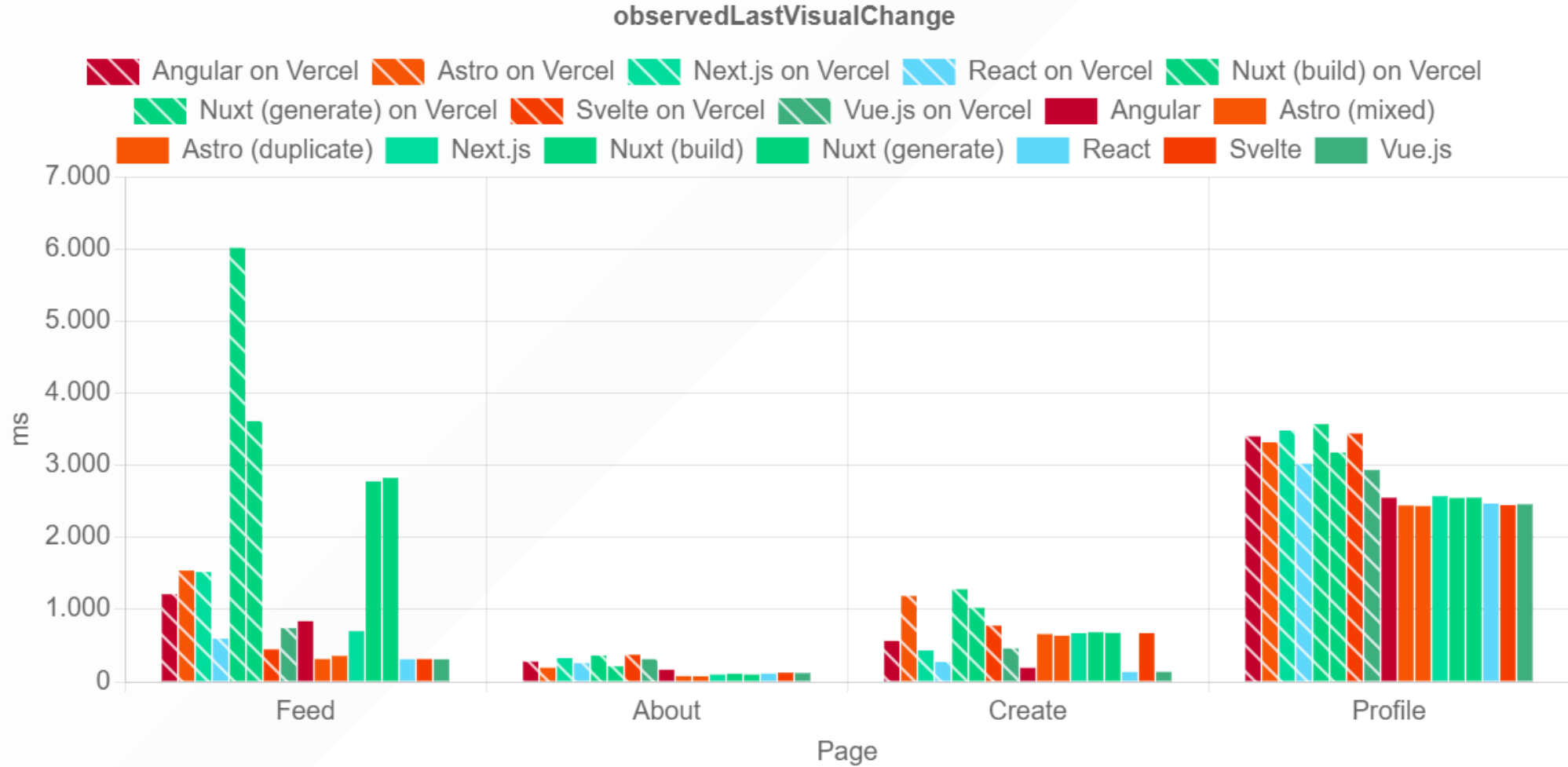


4.2 Component Load Time - OVCD

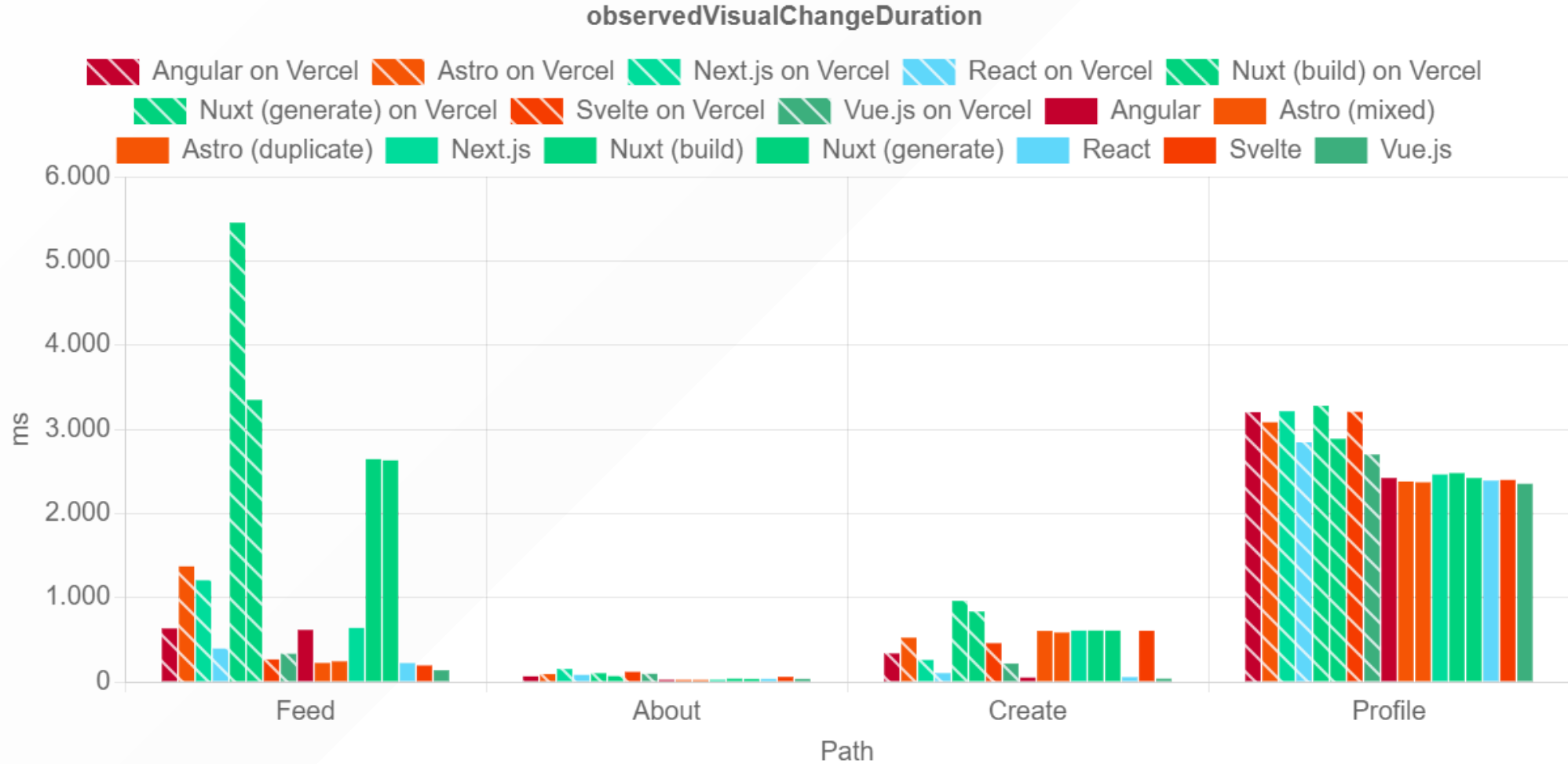
First und Last Visual Change umfassen auch die Datenübertragungszeit. Die Observed Visual Change Duration beschreibt die Zeit zwischen Anfang und Ende der visuellen Änderungen.

$$\textit{observedVisualChangeDuration} = \textit{OLVC} - \textit{OFVC}$$

4.1 Page Load Time - OLVC



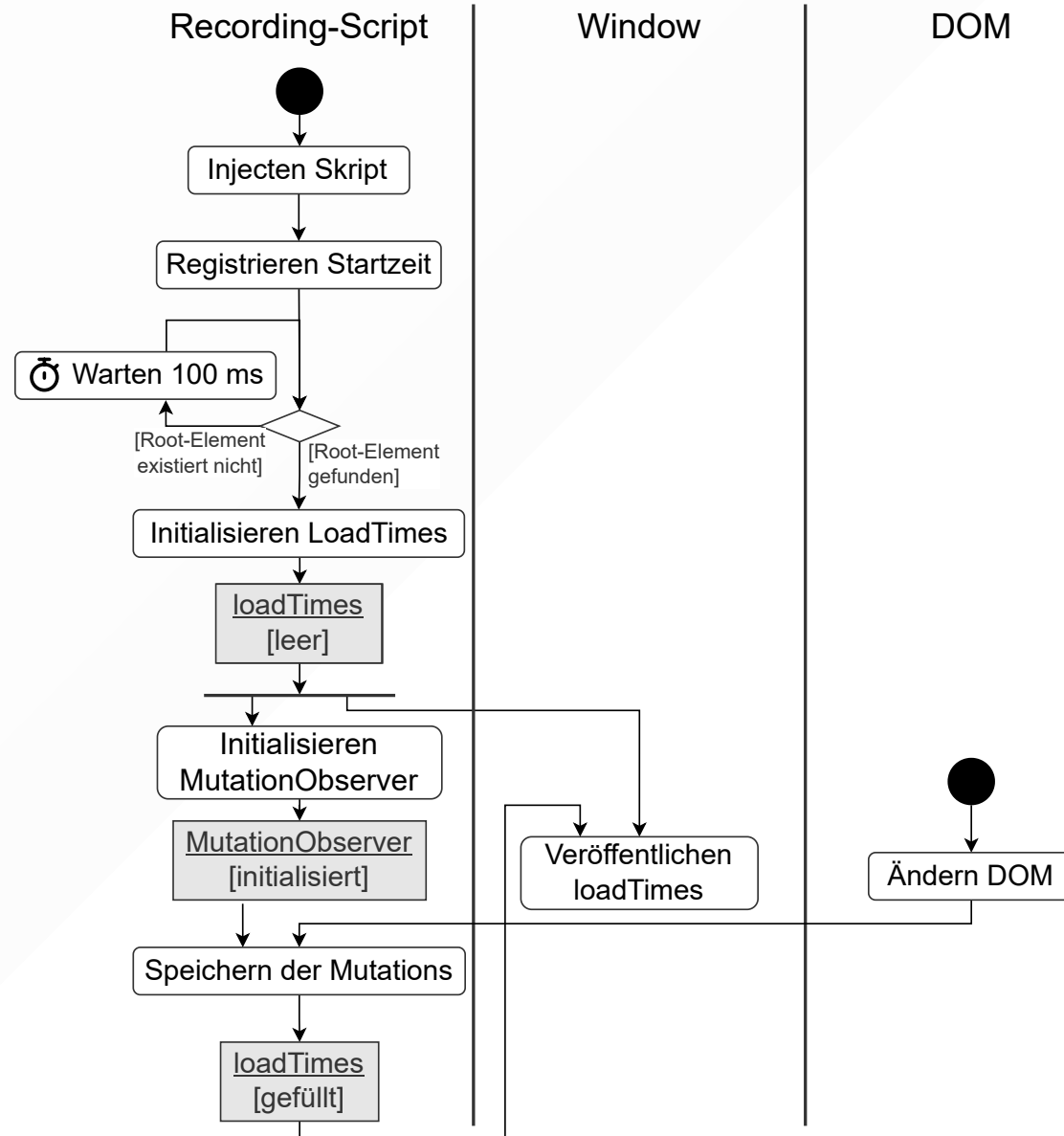
4.2 Component Load Time - OVCD



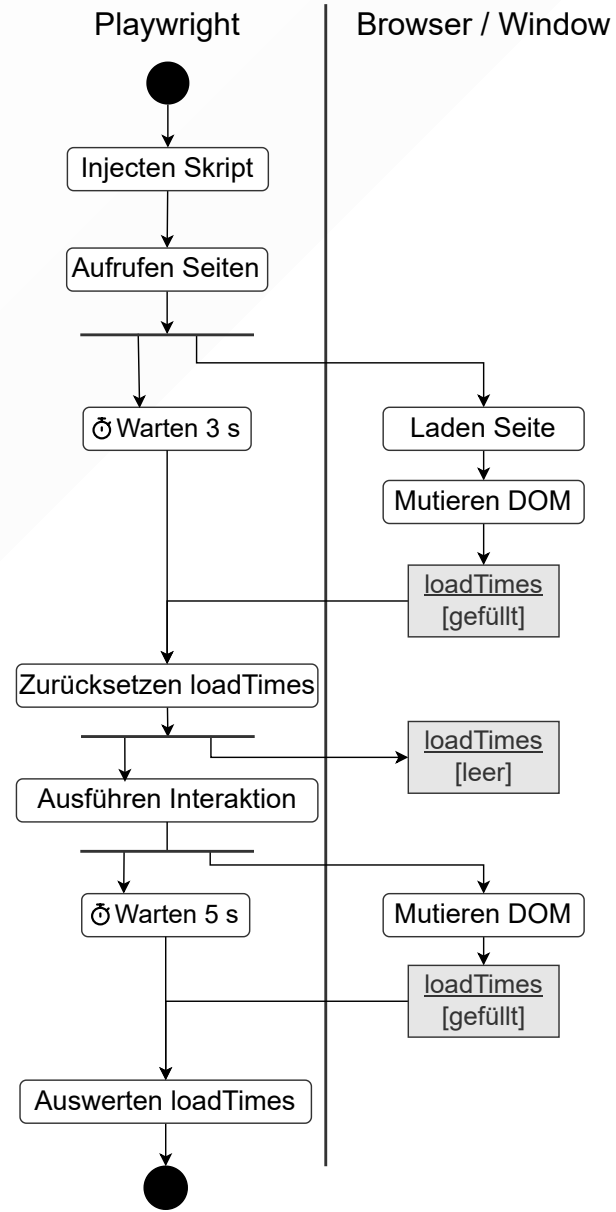
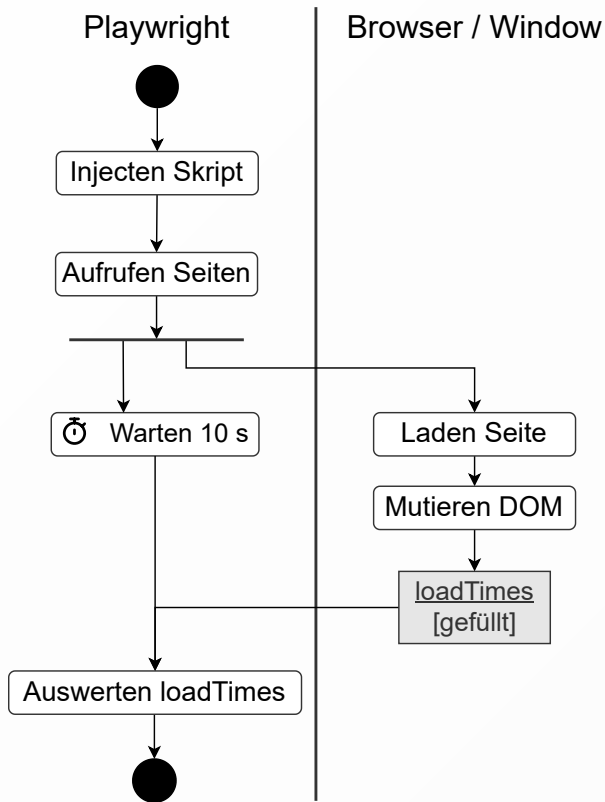
DOM Mutation Times

- Component Load Times
- Component Update Times

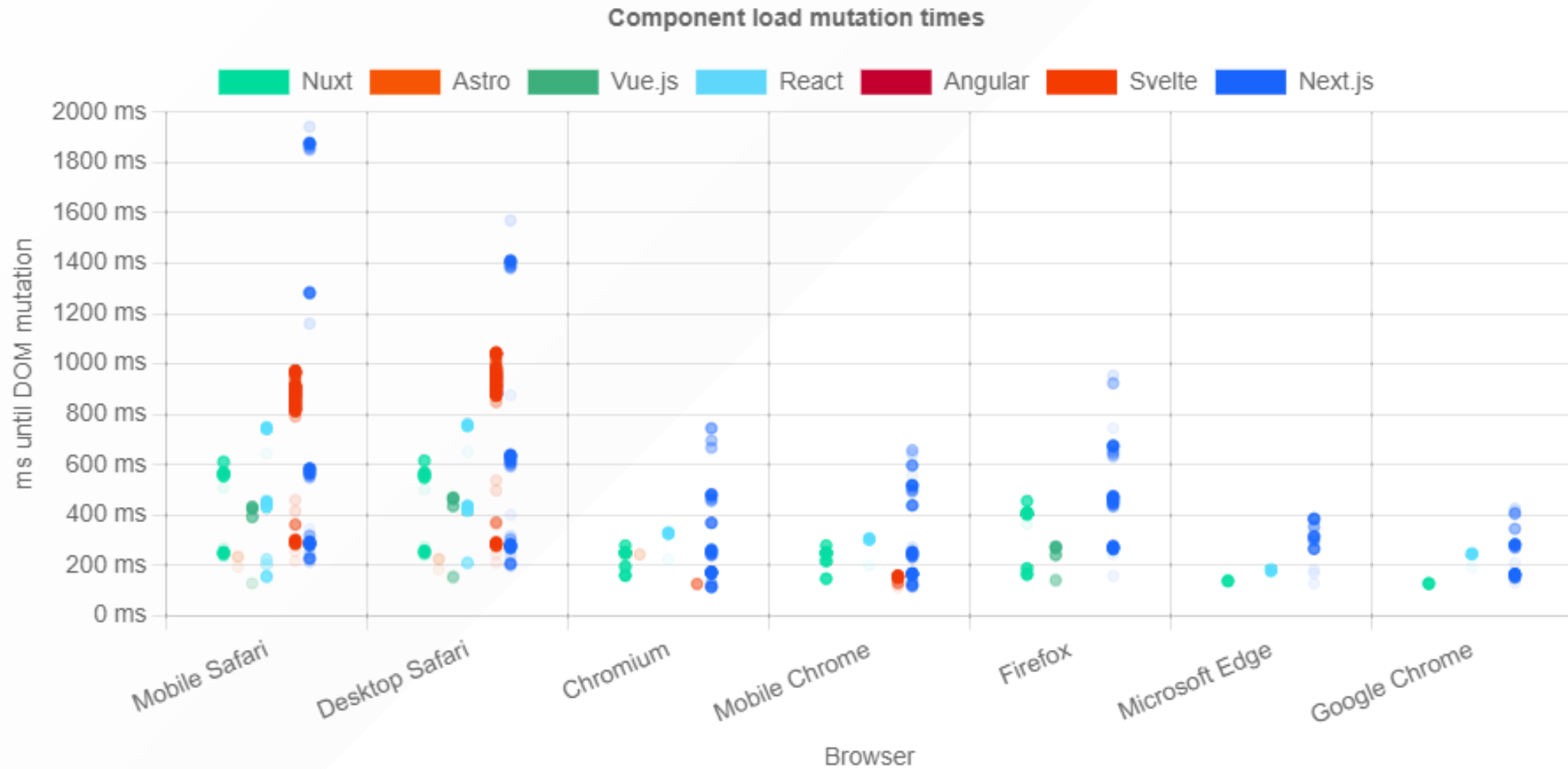
Aufzeichnung



Nutzung



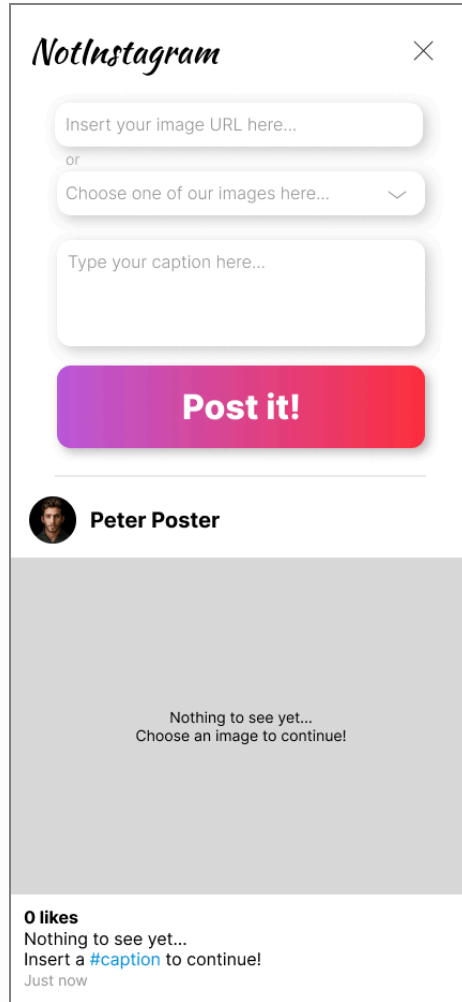
4.2 Component Load Times - DOM Mutations



4.2 Component Load Times - DOM Mutations

- Zwei Aufzeichnungsgrenzen
 - Initialisierungsintervall von 100 ms
 - festes Ende nach 10 s
- Fehlende Aufzeichnungen
 - Schnelle Updates beim Laden des DOMs
 - Langsame Updates nach Ende der Aufzeichnung

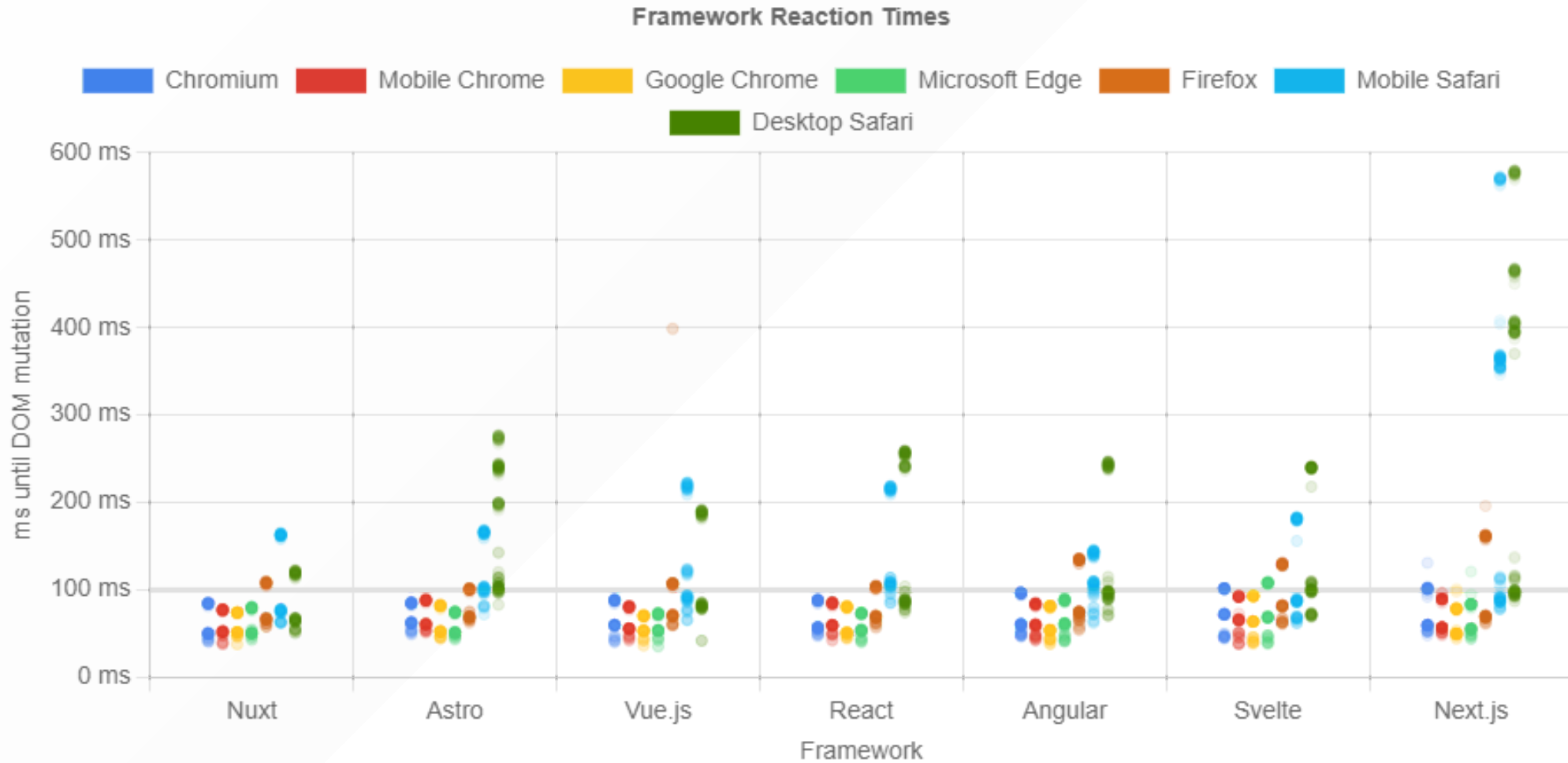
4.3 Component Update Times



The screenshot shows a mobile-style interface for creating a post. At the top, it says "NotInstagram" with a close button. Below are two input fields: "Insert your image URL here..." and "Choose one of our images here..." with a dropdown arrow. A third field is "Type your caption here...". A large red button labeled "Post it!" is below the caption field. Underneath is a profile section for "Peter Poster" with a circular profile picture. The main content area is a grey rectangle with the text "Nothing to see yet... Choose an image to continue!". At the bottom, it shows "0 likes", "Nothing to see yet...", "Insert a #caption to continue!", and "Just now".

1. Caption Insert
2. Media Selection
3. Source Insert
4. Post Creation (1. & 2.)

4.3 Component Update Times



4.3 Component Update Times - Messungen

ms	Nuxt	Angular	Vue.js	React	Astro	Svelte	Next.js	Ø
	51	77	47	84	87	70	79	71
Desktop Safari	86	123	136	169	170	164	304	164
	124	172	200	280	270	283	493	260
	47	52	52	67	78	56	73	61
Mobile Safari	110	106	133	126	154	126	196	136
	167	152	206	183	254	208	372	220
	59	54	52	54	63	60	59	57
Firefox	83	89	82	84	99	94	142	96
	108	123	103	181	142	129	235	146

4.3 Component Update Times - Messungen

ms	Nuxt	Angular	Vue.js	React	Astro	Svelte	Next.js	Ø
	42	44	46	44	49	45	47	45
Mobile Chrome	61	67	69	67	69	81	94	73
	82	90	89	82	85	116	143	98
	39	44	51	44	51	38	47	45
Chromium	66	69	77	58	71	74	75	70
	94	95	104	85	89	95	108	96

4.3 Component Update Times - Messungen

ms	Nuxt	Angular	Vue.js	React	Astro	Svelte	Next.js	Ø
	37	43	40	41	44	40	46	42
Microsoft Edge	61	70	61	62	64	74	73	67
	85	90	79	75	80	102	134	93
	34	41	37	40	43	39	41	39
Google Chrome	60	62	61	59	57	64	69	62
	77	84	77	77	72	89	99	82

4.3 Component Update Times - Messungen

ms	Nuxt	Angular	Vue.js	React	Astro	Svelte	Next.js
	44	51	46	53	59	50	56
Browser Average	75	84	88	89	98	97	136
	105	115	123	138	142	146	226
	35	45	36	45	48	42	45
Weighted Browser Average	60	69	70	75	74	78	107
	80	94	93	110	104	118	167

4.3 Component Update Times - Ranking

Frameworks

1. Nuxt
2. Angular
3. Vue.js
4. React
5. Astro / Svelte
6. Next.js

Browsers

1. Google Chrome
2. Microsoft Edge
3. Chromium
4. Mobile Chrome
5. Firefox
6. Mobile Safari
7. Desktop Safari

5. Lessons Learned

1. Ergebnisse
2. Methodik
3. Test-Ansatz für DOM-Mutationen

5.1 Ergebnisse

- Testergebnisse sind **nicht eindeutig** bzgl. Page Load Times und Component Load Times.
 - Component Update Times zeigen **undeutliche Tendenzen** auf.
-

CUT:

- Frameworks: Ø 69 - 107 ms
- Browsers: Ø 62 - 164 ms

5.2 Methodik

- Messergebnisse schwanken um bis zu 30%
- Verteilung der Ergebnisse könnte Performanceunterschiede aufzeigen
- Testumfang muss ausgeweitet werden
 - Seiten
 - Komponenten
 - Hosting Services
 - Test Runs

5.3 Test-Ansatz für DOM-Mutationen

- Aufzeichnungen von DOM-Mutationen fehlen am Anfang und Ende der Tests
 - White-Box Testing, um...
 - Aufzeichnungen zu triggern und...
 - Rendering-Ende zu signalisieren
-
- keine Tests zu Navigation zwischen Seiten

Dankeschön!

Mega-fast or just super-fast? Performance differences of mainstream JavaScript frameworks for web applications

Andreas Nicklaus, 44835

17.10.2024



Anhang

Referenzen

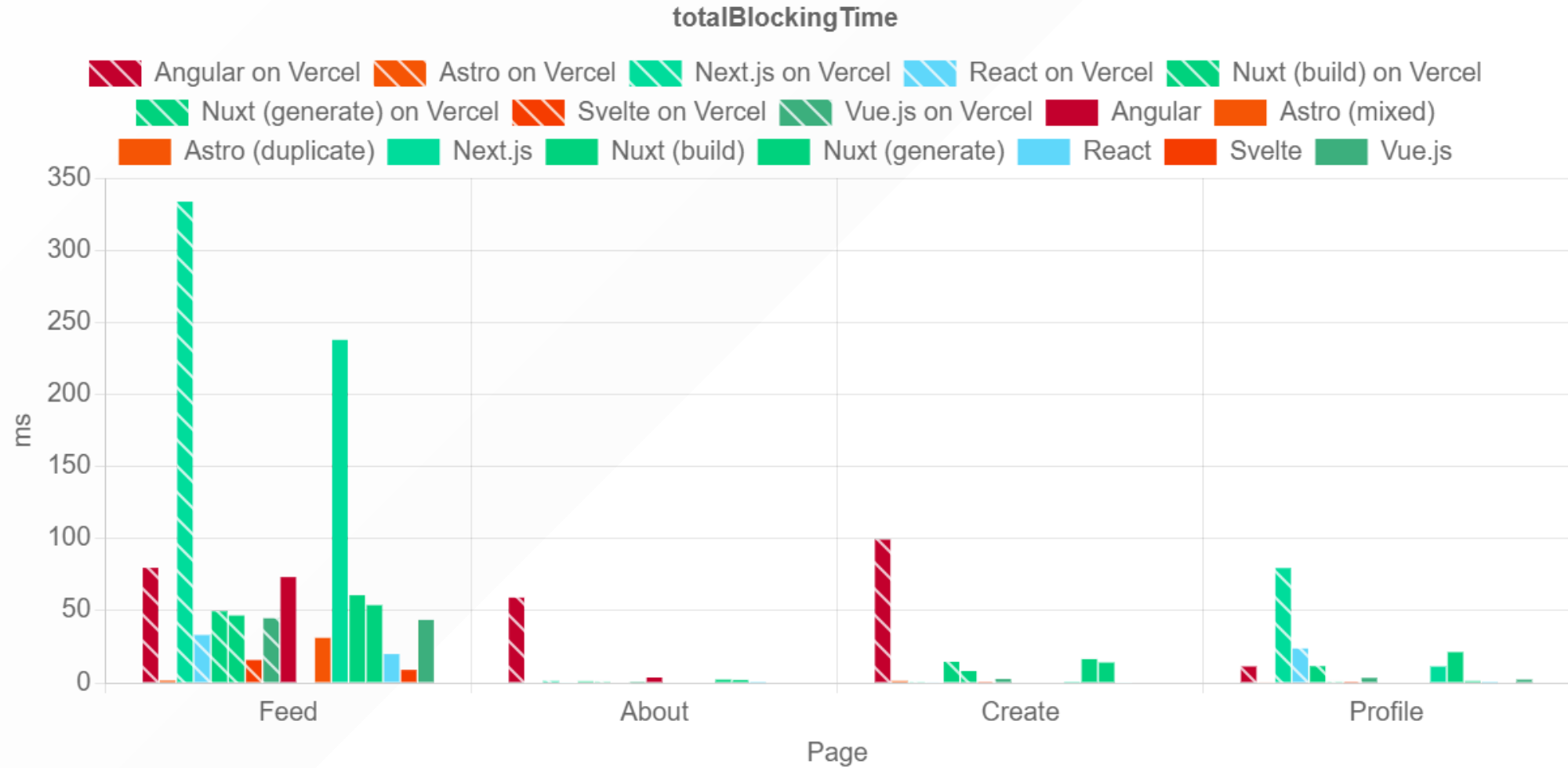
- [1] <https://www.scribd.com/document/471812575/A-website>
- [2] <https://2023.stateofjs.com/en-US/libraries/front-end-frameworks/>
- [3] <https://vercel.com/>
- [4] <https://developer.chrome.com/docs/lighthouse/overview>
- [5] <https://playwright.dev/>
- [6] <https://gs.statcounter.com/>

Abkürzungsverzeichnis

Abkürzung	Bedeutung
CI/CD	Continuous Integration and Continuous Delivery
CSR	Client-Side Rendering
DOM	Document Object Model
FVC	First Visual Change
HTML	Hypertext Markup Language
JS	JavaScript
LCP	Largest Contentful Paint
LVC	Last Visual Change

Abkürzung	Bedeutung
OLVC	Observed Last Visual Change
OFVC	Observed First Visual Change
OVCD	Observed Visual Change Duration
SSR	Server-Side Rendering
TBT	Total Blocking Time
TBW	Total Byte Weight
TTFB	Time To First Byte
TTI	Time To Interactive

4.1 Page Load Time - TBT



4.1 Page Load Time - LCP

