rxAngular

Endbericht | Call 18 | Projekt ID 6798

Lizenz: CC BY-SA

Inhalt

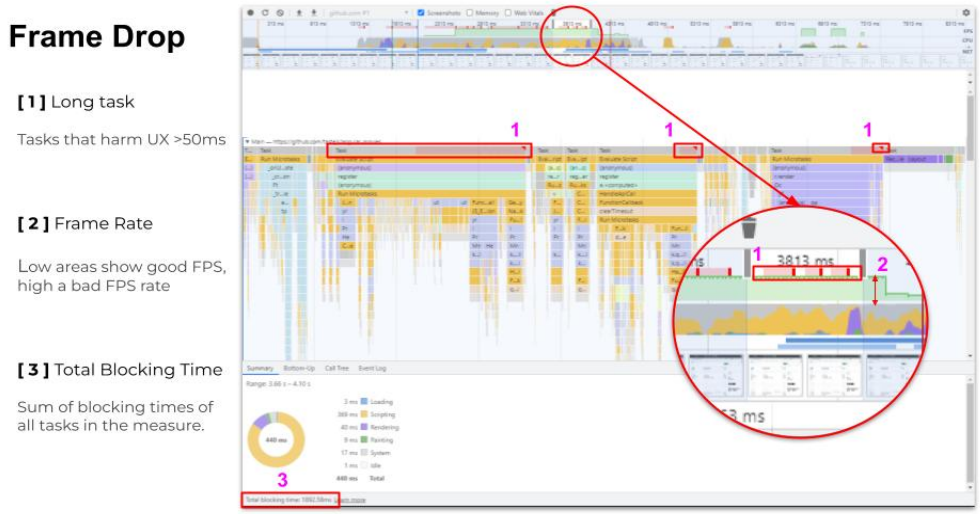
1 Einleitung

Als Datenströme bezeichnet man in der Informatik einen kontinuierlichen Fluss von Datensätzen, dessen Ende meist nicht im Voraus abzusehen ist. Die Datensätze werden fortlaufend verarbeitet.
Datenströme treten in unzähligen und häufig (zeit-)kritischen Systemen weltweit auf, z.B. Überwachung von Infrastruktur, Aktienhandel, Medizingeräten (z.B. EKG-Gerät).
Aufgrund beschränkter Ressourcen auf Verarbeitungs- oder Anzeigegeräten sind intelligente Verarbeitungsmethoden und -algorithmen nötig, um Verzögerungen bei der Bearbeitung/Anzeige so gering wie möglich zu halten.

Obwohl moderne CPUs Multi-Threading-fähig sind, beschränken browserbasierte Anwendungen jedes Fenster bzw. jedes TAB auf einen einzigen Thread ("Single-Thread-Umgebungen"). Derzeit werden browserbasierte Anwendungen für Benutzeroberflächen in einer Vielzahl von Systemen verwendet, z. B:

- Desktop (Browser, hybride Anwendungen)
- Mobiltelefone (Browser und Apps)
- Tablets (Browser und Apps)
- Medizinische Geräte in Krankenhäusern
- Touchscreens in Autos
- Jede Umgebung, die JavaScript ausführen kann (z.B. Raspberry Pie, Arduino und andere Chips)

Das folgende Bild zeigt Tasks, die in einer Single-Thread-Umgebung seriell abgearbeitet werden. Dieses konkrete Beispiel zeigt einen Browser. Dieser besitzt nur einen einzigen UI-Thread (Haupt-Thread), d. h. die durchzuführenden Arbeiten laufen nacheinander ab. Unter jedem Task ist die Arbeitslast innerhalb dieses Tasks angegeben.
Jeder rot markierte Task ist ein sogenannter “Long Task” (>50ms), der das Zeitbudget pro Frame (“Frame Budget”) von 50ms überschreitet. Diese Long Tasks verhindern eine kontinuierliche Visualisierung, d.h. einzelne Frames müssen ausgelassen werden und werden nicht angezeigt (Frame Drops). Dies führt in Folge zu Verzögerungen, Lags oder Freezes und im schlimmsten Fall zu Informationsverlust. Für Benutzende stellt dies schlechte Benutzererfahrung dar.



Mit Stand der Technik führt hohe Arbeitslast in browserbasierten Anwendungen meist unvermeidlich zu Frame Drops und somit schlechter User Experience. Dies gilt insbesondere für mobile Geräte, da sie im Vergleich zu stationären Geräten über eine geringere Rechenleistung verfügen (begrenzte Ressourcen bei

Prozessorleistung und Grafikleistung). Mobile Geräte sind auch anfällig für eine schlechtere Konnektivität. Es werden hoch optimierte Datenströme benötigt.

Um die Probleme von Frame Drops bzw. Informationsverlust zu vermeiden, müssen Datenströme so schnell wie möglich verarbeitet und ggf. optimiert werden.
3 generelle Schlüsselkonzepte:

1. Reduce work / Arbeit reduzieren
2. Distribute work / Arbeit verteilen/aufteilen
3. Prioritize work / Priorisierung der Arbeit

Schnelle und optimierte Datenströme sind vor allem für Real-time Collaboration über das Internet erforderlich:

- Umgebungen, in denen mehrere Benutzer gemeinsam und gleichzeitig an einem Objekt arbeiten
- kollaborative Online-Mindmaps
- kollaborative Online-Dokumente
- Online-Code-Editoren

oder Online-Kommunikation:

- Social Media Feeds
- Standortfreigabe
- Nachrichtenübermittlung / Messengers

Stand der Technik (zu Projektbeginn)
Bestehende Technologien für die Verarbeitung von Datenströmen sind User Space Scheduler.
Ein Scheduler ist eine komplexe Software, die dafür sorgt, dass die CPU-Kerne nicht ungenutzt bleiben, wenn es Threads gibt, die bearbeitet werden müssen. Der Scheduler sorgt für den schnellen Wechsel zwischen den Threads und erweckt den Eindruck, dass alle Prozesse parallel ablaufen.

Unzulänglichkeiten von User Space Schedulern:

- Messungen zum Frame Budget: Es gibt keine Möglichkeit, die (noch) benötigte Ausführungszeit eines Tasks während der Laufzeit zu messen. Es ist lediglich im Nachhinein bekannt, wenn ein Task sein Frame Budget überschritten hat (zu spät!).
- Eine Priorisierung der Arbeit innerhalb eines Tasks ist nicht möglich.

Mit den vorhandenen Lösungen ist es nur möglich, die Ausführung von Browser-Tasks zu verzögern, aber es gibt keine Möglichkeit für die Verteilung von Arbeit, welche erforderlich wäre, um das Frame Budget und die Prioritäten des Browsers optimal zu erfüllen.
Es gibt keine Lösung für die Verbindung von Browser-Task-Scheduling mit Priorisierung und Frame-Budget-Metrik.

2 Projektbeschreibung

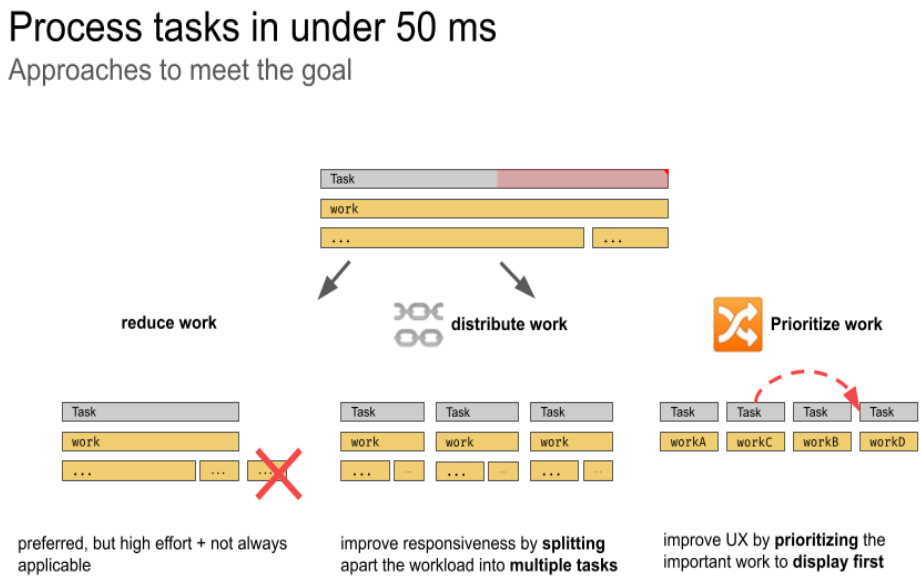
Generelles Entwicklungsziel

Generelles Entwicklungsziel sind neuartige Verfahren zur Beschleunigung der Verarbeitung und Anzeige von Daten auf stationären und mobilen Webclient-Endgeräten mittels neuartiger Algorithmen zur unterbrechungsfreien und intelligenten Aufteilung und Priorisierung von Datenströmen sowie die Untersuchung der Anwendbarkeit der neu entwickelten Verfahren in bestehenden Webtechnologien.

Unsere Verfahren sollen auf 3 grundlegenden Ansätzen basieren:

1) Arbeit reduzieren. Ziel ist, dass Benutzerinteraktionen und Hintergrundprozesse gemeinsam ablaufen, ohne sich gegenseitig zu behindern.
   Unser Ansatz: Gesamtausführungszeit verkürzen durch z.B. intelligentes Löschen
2) Arbeit aufteilen/verteilen. Ziel ist, die Gesamtmenge der Arbeit in ungefähr gleiche, nicht blockierende Tasks aufzuteilen, die ihre maximale Ausführungszeit ("Frame Budget") einhalten und den Benutzer nicht von anderen Interaktionen (z. B. mit der Schnittstelle der Anwendung) abhalten.
3) Arbeit intelligent priorisieren. Ziel ist, dass die Entwickler die Behandlung von Benutzerinteraktionen oder Hintergrundprozessen orchestrieren können, indem sie ihnen Prioritäten zuweisen.

Erforscht werden müssen die intelligente Kombination dieser Ansätze sowie Entscheidungskriterien für deren optimale Anwendung.



Zielgruppen

Zielgruppe 1: Betreiber:innen von E-Commerce-Websiten
Nutzende erwarten, dass Webseiten in <3s geladen sind, sonst brechen 53% den Ladevorgang ab (Google 2017) - eine direkte Auswirkung von schlechter UX aufgrund von Performancemängeln. Umgekehrt: Bei Ladezeitverringerungen von nur 0,1s steigen Konversionsraten im Schnitt um 8% bei Einzelhandels-Websites und um 10% bei Reise-Websites (Deloitte 2020).

Zielgruppe 2: Entwickler:innen und Anbieter:innen großer Webplattformen
Bedürfnis der Zielgruppe: Schnelle Anzeige von Daten und unterbrechungsfreie Interaktion mit Web-Content, insbesondere wenn die Webanwendungen sicherheitsrelevante Informationen verbreiten oder allgemein schnell zugänglich sein müssen (z.B. Wetterinformation, Nachrichten).

Zielgruppe 3: Angular Entwickler
Bedürfnis der Zielgruppe: Schnellere Zugriffe und schnelle Verarbeitung von Daten; Verbesserung von Runtime Performance, im speziellen Verbesserung der neuen Web-Performance-Metrik INP ("interaction to next paint")

Überblick über das Projektergebnis

Alleinstellungsmerkmal: Erstmalige Kombination von Browser-Task-Scheduling mit Priorisierung und Bestimmung der Frame-Budget-Metrik.

Weiterentwickelt wurde ein Modul zur *Frame-Budget-Optimierung*. Im Unterschied zum Stand der Technik ist es damit nicht erst nach Prozessende, sondern erstmals bereits zur Runtime möglich, das bereits verbrauchte Zeit-Budget eines Tasks zu ermitteln und so Maßnahmen zur Steuerung zu ergreifen (z.B. bei aufwendigem Listen-Rendering).

Weiterentwickelt wurde ein Software-Modul für *Task Abortion*. Mit bisherigen Lösungen vom Stand der Technik bestand hohes Risiko, dass unnötige Arbeit ausgeführt wird, die für die Benutzenden nicht sichtbar ist (z.B. das Aktualisieren von Elementen, die im nächsten Task gelöscht werden). Dies wird mit unserer Lösung verhindert, indem derartige Tasks automatisiert erkannt und aus der Task Queue entfernt werden.

Weiterentwickelt wurde ein Software-Modul für intelligentes Coalescing mit neuartigem Scoping-Mechanismus. Eine zu rendernde Komponente wird damit mithilfe eines neu entwickelten Scoping-Algorithmus analysiert und mit einem Scoping-Flag (Markierung) versehen, damit sichergestellt wird, dass ausschließlich relevante Elemente für das nächste Render-Update herangezogen werden.

3 Verlauf der Arbeitspakete

3.1 Arbeitspaket 1 - *Detailplanung und Formales am Projektstart*

Haupttätigkeiten:

- Verfeinerung der Entwicklungs-Roadmap
- 1. Blogbeitrag verfassen
- 1. Förderrate beantragen

Hauptergebnis: 1. Blogbeitrag
https://www.netidee.at/rxangular/rxangulars-roadmap-after-funding

Abweichungen zum Plan: 1 neues Team-Member
Mit Enea Jaholari kam ein weiterer Entwickler zum Team, dessen Onboarding-Phase erst nach Antragsabgabe abgeschlossen war. Mit seinem Spezialwissen im Bereich des performantem List-Diffing kann er einen wichtigen Beitrag zum Projekt leisten.

3.2 Arbeitspaket 2 - *Entwicklung Modul Frame-Budget-Optimierung*

Haupttätigkeiten:

- Wesentliche Verbesserungen zur Virtual-Scrolling-Funktion
- Erster Entwurf zu einem Reconciliation-Algorithmus für effizienteres, asynchrones Listen-Diffing

Wir haben erkannt, dass die Virtual-Scrolling-Funktion (= Benutzer:innen wird nur ein kleiner Teil der Daten zu einem bestimmten Zeitpunkt angezeigt) mit Stand der Technik nur ineffizient genutzt wird:

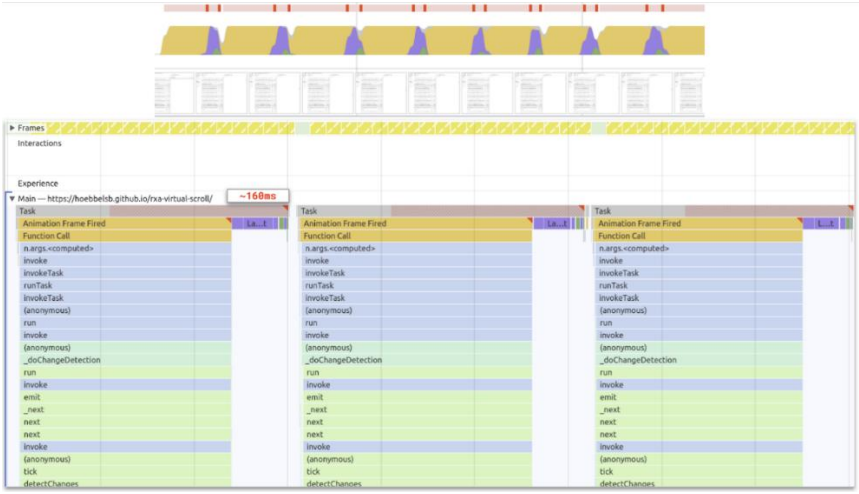
1) Die Anzeige der sichtbaren Elemente hatte zur Folge, dass die Position anderer Elemente invalidiert wurde und bei jedem Update neu berechnet werden musste.
2) Ineffizienz, da die Updates beim Scrollen auf dem Main-Thread ausgeführt wurde.

Unsere Verbesserungen beruhen auf einer Kombination von 3 verschiedenen Ansätzen:

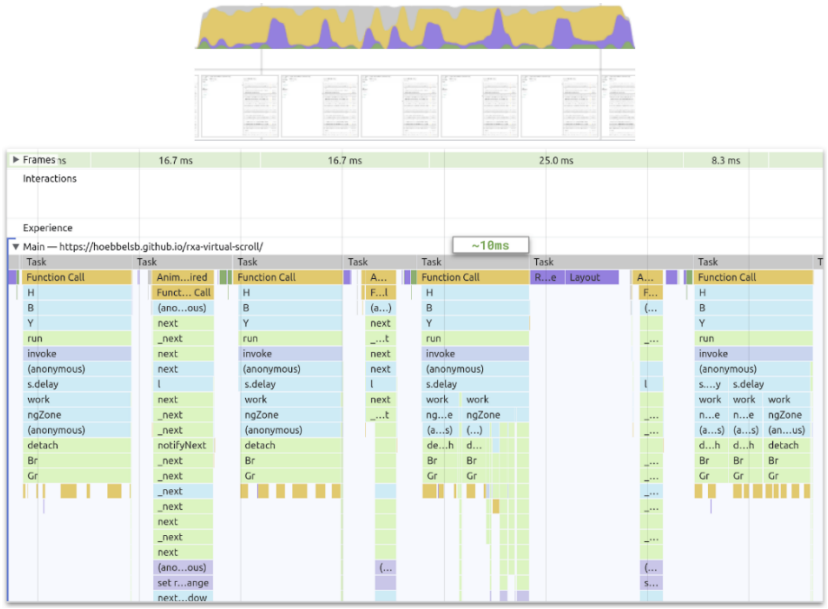
1) Gänzliches Weglassen aller Elemente, die nicht sichtbar sind
2) Bei Einblendung neuer Elemente die Anzeigestruktur alter Elemente recyclen, anstatt neu zu erzeugen
3) Feinkörnige Reaktivität: Nur Elemente, welche tatsächlich verändert werden müssen, werden auch neu dargestellt.
4) Concurrent Mode: Anwendung des Concurrent Modes auf sichtbare Elemente.

Messergebnisse:
https://www.rx-angular.io/docs/template/api/virtual-scrolling#comparison-with-angular-cdk

Existierende Virtual-Scrolling-Lösung: 160ms **blocking time**



Unsere Lösung: 10ms; **no blocking time**



Demo-Applikation zum Vergleich verschiedener Scroll-Strategien:
https://hoebbelsb.github.io/rxa-virtual-scroll/

Abweichungen zum Plan: Ausfall zweier Team-Member; Verlängerung um 3 Monate

NEU (seit dem Zwischenbericht)

Neuer Reconciliation-Algorithmus

https://github.com/rx-angular/rx-angular/pull/1825

Unsere bisherige Lösung war, den mit Stand der Technik schnellsten Listen-Diffing-Algorithmus („IterableDiffer“) mit unserem Chunking-Algorithmus zu kombinieren. Unser Ansatz dabei war, die im Diffing-Algorithmus durchgeführten 4 logischen Gruppierungen (Item Add, Item Delete, Item Move, Item Update) in einzelne Arbeitspakete aufzuteilen.
Das Problem des IterableDiffers ist die Anzahl der Operationen. Besonders, bei `move` oder `swap` diffs, berechnet der IterableDiffer eine unnötig hohe Anzahl Operationen. Dies hat zur Folge, dass unnötig viel Aufwand mit dem Neuzeichnen von Templates verbracht wird und somit die Performance nicht optimal ist.

Mit Angular Version 17 wurde ein neuer Listen-Diffing-Algorithmus implementiert: der reconciliation algorithm. Dieser adressiert die Probleme des IterableDiffers und produziert die optimale Anzahl an Operationen für alle Operationstypen: Add, Move, Delete.
Das Problem ist jedoch, dass dieser interne Angular-Reconciliation-Algorithmus synchron arbeitet und somit nicht mit unserem Konzept “Chunking” vereinbar ist. Wir mussten den bestehenden Algorithmus also dahingehend weiterentwickeln, sodass wir Operationsketten asynchron über unsere Render-Strategien abarbeiten können.

Die folgende Tabelle zeigt das Ergebnis eines Benchmarks, welches die Performance des IterableDiffers mit dem neuen Reconciliation-Algorithmus vergleicht.
`angular-ngfor-v17.0.0-rc.0` (4.Spalte) steht für den IterableDiffer Algorithmus. Die Spalten `angular-cf-*` (Spalten 1-3) zeigen den neuen Reconciliation-Algorithmus in verschiedenen Framework-Modi.
Zu Betonen ist hier die Reihe `swap rows`, welche die Performance Unterschiede bei Swap-Operationen deutlich herausstellt.

Name Duration for...	angular-cf-nozone-v17.0.0-rc.0	angular-cf-v17.0.0-rc.0	angular-cf-signals-v17.0.0-rc.0	angular-ngfor-v17.0.0-rc.0
Implementation notes				
Implementation link	code	code	code	code
create rows creating 1,000 rows (5 warmup runs).	45.2 ±0.5 (1.15)	46.0 ±0.4 (1.17)	47.1 ±0.4 (1.20)	47.6 ±0.3 (1.21)
replace all rows updating all 1,000 rows (5 warmup runs).	47.7 ±0.7 (1.20)	49.8 ±0.6 (1.25)	49.7 ±0.6 (1.25)	50.0 ±0.7 (1.26)
partial update updating every 10th row for 1,000 rows (3 warmup runs). 4 x CPU slowdown.	22.3 ±0.9 (1.12)	21.8 ±0.6 (1.10)	22.2 ±0.5 (1.11)	22.7 ±0.6 (1.14)
select row highlighting a selected row. (5 warmup runs). 4 x CPU slowdown.	4.7 ±0.2 (1.43)	4.8 ±0.2 (1.48)	7.6 ±0.2 (2.32)	5.0 ±0.2 (1.54)
swap rows swap 2 rows for table with 1,000 rows. (5 warmup runs). 4 x CPU slowdown.	24.8 ±0.4 (1.08)	25.4 ±0.5 (1.11)	25.7 ±1.2 (1.12)	166.8 ±1.1 (7.28)

Source: https://blog.angular.dev/introducing-angular-v17-4d7033312e4b

RxVirtualView

https://github.com/rx-angular/rx-angular/pull/1819

https://github.com/rx-angular/rx-angular/pull/1816

In Q4/2024 haben wir an dem neuen Feature RxVirtualView gearbeitet und im Dezember veröffentlicht (noch als developer preview).

Was ist RxVirtualView?

Ein Feature, um nur jene DOM-Knoten anzuzeigen, die für den Benutzer tatsächlich sichtbar sind. Es erweitert die virtuelle Scroll-Funktion mit einem generischen Ansatz, wenn es sich bei dem anzuzeigenden Content nicht um eine einfache Liste handelt.

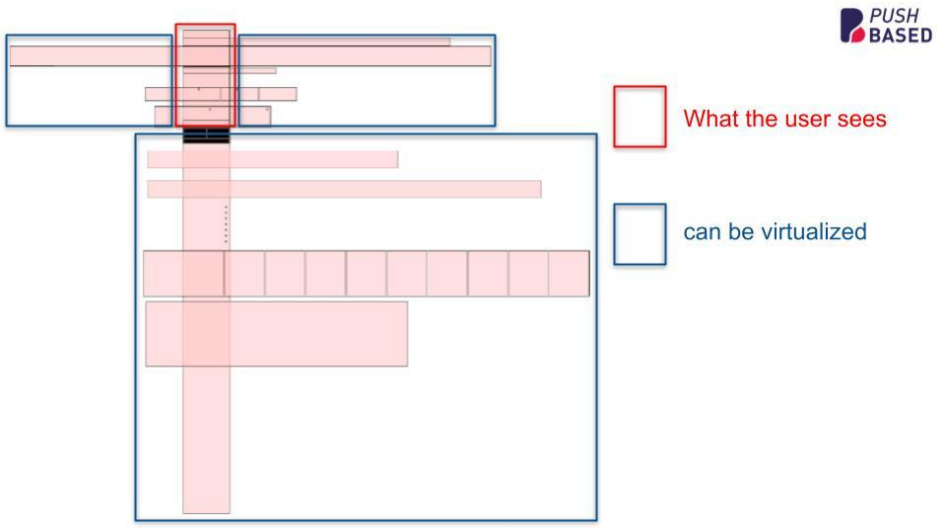
Motivation

Eine große Anzahl von DOM-Elementen kann die Leistung erheblich beeinträchtigen, was zu langsamen Anfangsladezeiten und trägen Interaktionen führt.
Vor allem mobile Nutzer haben einen sehr begrenzten Sichtbereich zur Verfügung. Die meisten Seiteninhalte sind unterhalb des Folds verborgen. Warum sollten sie also überhaupt gerendert werden?

Beim Umgang mit großen Listen oder Datensätzen gibt es eine Technik, die als virtuelles Scrollen oder Windowing bekannt ist. Dadurch wird die Leistung Ihrer Angular-Anwendungen drastisch verbessert. Wenn Sie jedoch nicht mit einfachen Listen oder hochdynamischen Komponenten arbeiten, ist das Konzept des virtuellen Scrollens nicht anwendbar. Dies gilt für:

- masonry layouts
- dynamic grids
- landing pages with widgets

An dieser Stelle kommt die RxVirtualView-Direktive ins Spiel. Sie bietet eine einfache Möglichkeit, nur die Elemente anzuzeigen, die derzeit für den Benutzer sichtbar sind.



Abweichungen zum Plan: Ausfall zweier Team-Member; Verlängerung um 3 Monate

3.3 Arbeitspaket 3 - *Entwicklung Modul Task Abortion*

Haupttätigkeiten: Weiterentwicklung für Native PostTask-Scheduling-Support

Unsere Entwicklungstätigkeiten zielten darauf ab, die bestehenden Nachteile von benutzerdefinierten Schedulern zu lösen: Während es möglich ist, einen benutzerdefinierten Scheduler zu erstellen, um Aufgaben innerhalb der von Ihnen kontrollierten Codebasis zu priorisieren, verwenden Skripte von Drittanbietern möglicherweise nicht denselben Scheduler. Folglich wird die Priorisierung von Aufgaben in Umgebungen schwierig, in denen Sie nicht die Kontrolle über den gesamten Code auf der Seite haben. Die Alternativen beschränken sich auf das Aufteilen von Aufgaben oder das explizite Nachgeben gegenüber Benutzerinteraktionen. rxAngular zielt darauf ab, diese Nachteile zu überwinden, indem wir die native scheduler.postTask-API einbinden (in Arbeit).

Weitere Arbeiten:

- Task Aborting: Erweiterung der Implementierung zur Unterstützung des Aufgabenabbruchs, so dass geplante Aufgaben abgebrochen werden können.
- Task Prioritization Configuration: Untersuchung von Optionen zur Konfiguration der Aufgabenpriorisierung auf der Grundlage spezifischer Anforderungen oder Anwendungsbedürfnisse.
- Testing: Implementierung von Tests zur Sicherstellung der Zuverlässigkeit und Kompatibilität der nativen scheduler.postTask-API-Integration in verschiedenen Browsern und Umgebungen.

NEU (seit Zwischenbericht)

https://github.com/rx-angular/rx-angular/pull/1843 (feat: make scheduler features configurable)

Wir haben unseren internen Scheduler weiterentwickelt, um ihn für unsere Anwender:innen konfigurierbar zu machen. Darüber hinaus haben wir eine neue Funktion entwickelt und implementiert, welche es ermöglicht das experimentelle `Scheduler.isInputPending`-Feature[1] zu verwenden. Es ermöglicht zu überprüfen, ob es ausstehende Eingabeereignisse in der Ereignis-Warteschlange gibt, was darauf hinweist, dass der Benutzer versucht, mit der Seite zu interagieren.

Diese Funktion ist in Situationen nützlich, in denen eine Warteschlange von Aufgaben besteht, die ausgeführt werden sollen, und regelmäßig dem Haupt-Thread Vorfahrt einräumen möchten, um die Benutzerinteraktion zu ermöglichen, damit die Anwendung so reaktionsschnell und leistungsfähig wie möglich bleibt.

Somit können wir unsere durch den "Chunking' Algorithmus erstellte Arbeits-Warteschlange pausieren, sobald Benutzer:innen wieder mit der Seite interagieren möchten.

Abweichungen zum Plan: Ausfall zweier Team-Member; Verlängerung um 3 Monate

[1] https://developer.mozilla.org/de/docs/Web/API/Scheduling/isInputPending

3.4 Arbeitspaket 4 - *Entwicklung Modul Coalescing mit Scoping-Mechanismus*

Haupttätigkeiten: Weiterentwicklungen zur Einbindung von Angular Signals.
Signals sind insbesondere für Junior Angular-Entwickler*innen ein hilfreiches Tool, da sie das reaktive Programmieren entscheidend erleichtern. Im Grunde sind sie ein Signalgeber mit einem Wert, der alle verbundenen Konsumenten benachrichtigt, wenn sich dieser Wert ändert.

Wir haben erkannt, dass es mit Angular Signals zu unerwarteten Problemen (konkret: Over Rendering) mit einer unserer bisherigen Render-Strategie kommt, wenn Flags zur Markierung von zu rendernden Objekten eingesetzt werden. Als mögliche Lösungsansätze haben wir untersucht, ob und wie sich die Deaktivierung von Parent-Flags auswirkt.

Ergebnis: Wir haben diese Diskussion lange geführt und können nun endlich damit beginnen, das Parent-Flag in den Render-Strategien und als Eingabe für unsere Direktiven zu verwerfen. Mit den neuen Signal Queries ist das Parent-Flag überflüssig, es verursacht nur unnötige Änderungserkennungszyklen. Dies wurde mit neuen Tests und einer neuen Demo bestätigt.

deprecate parent flag: https://github.com/rx-angular/rx-angular/pull/1710
drop parent flag: https://github.com/rx-angular/rx-angular/pull/1734
signal compatability: https://github.com/rx-angular/rx-angular/pull/1714

NEU (seit Zwischenbericht)

https://github.com/rx-angular/rx-angular/pull/1832 (feature added: provideRxRenderStrategies provider function)
Mit dem Upgrade von Angular ergaben sich auch für unsere Funktionen neue Möglichkeiten zur Verbesserung der Benutzbarkeit: Wir haben eine neue Funktion entwickelt, mit der die Render-Strategien, welche das Konzept für Coalescing und Scoping abstrahieren, in einfacherer Weise den Benutzenden zur Verfügung gestellt werden kann. Überdies wird ebenso die Erweiterbarkeit/Individualisierbarkeit wesentlich verbessert. Es ist nun einfacher für Entwickler:innen eigene Render-Strategien zu entwickeln (über unsere Strategievorlagen hinaus).

https://github.com/rx-angular/rx-angular/pull/1814 (fix: replace toObservableMicrotask private API with proper solution)
Notwendige Anpassung des internen Codes. Nachdem unsere Direktiven und Services bereits kompatibel mit den Angular Signals sind, unterliegen Sie auch Veränderungen des Frameworks, sollte dieses etwas an den internen Strukturen verändern. In Version 19 wurden die Timings der Effect-Scheduler verändert, welches sich ebenfalls auf unsere Direktiven auswirkte. Um unsere Lösungen weitestgehend robust zu gestalten, haben wir Teile des Schedulers selbst implementiert.

Abweichungen zum Plan: Ausfall zweier Team-Member; Verlängerung um 3 Monate

3.5 Arbeitspaket 5 - *Projektmanagement und Zwischenbericht*

Haupttätigkeiten:

- Zwischenbericht erstellen
- 2. Blogbeitrag verfassen
- 2. Förderrate beantragen

- 3. Blogbeitrag verfassen

Hauptergebnis: 2 Blogbeiträge

https://www.netidee.at/rxangular/new-features-rxangular (27.5.2024)
https://www.netidee.at/rxangular/halbzeit (03.09.2024)

Abweichungen zum Plan: keine

3.6 Arbeitspaket N - *Dokumentation und Formales am Projektende*

Haupttätigkeiten:

- Homepage, Dokumentation und Tutorials (neue Features) aktualisieren
- APIs entwickeln
- Weiterentwicklung auf Version 18 aufgrund Update des zugrunde liegenden Frameworks (Angular 18)
- 4. Blogbeitrag verfassen
- Endbericht verfassen, Entwickler:innen&Anwender:innen-Doku, Endabrechnung

Problem mit Meta-Tags gelöst, wenn unsere Homepage auf Social Media geteilt wird.
https://github.com/rx-angular/rx-angular/pull/1732

Angular18 Update (https://github.com/rx-angular/rx-angular/pull/1730)

- Deprecated API-Verwendungen verhindern
- Demo App wurde an neue Angular-Features angepasst
- Version bump & Anpassung von Peer-Dependencies:
    - @rx-angular/cdk
    - @rx-angular/template
    - @rx-angular/state
    - @rx-angular/isr

Abweichungen zum Plan: keine

NEU (seit Zwischenbericht)

https://github.com/rx-angular/rx-angular/pull/1824
https://github.com/rx-angular/rx-angular/pull/1793
https://github.com/rx-angular/rx-angular/pull/1848
Verbesserung des Entwicklerhandbuchs: Strukturverbesserungen, vereinfachte Navigation, bessere Code Beispiele

Angular 19 Upgrade (https://github.com/rx-angular/rx-angular/pull/1809)

- Deprecated API-Verwendungen verhindern
- Demo App wurde an neue Angular-Features angepasst
- Version bump & Anpassung von Peer-Dependencies:
    - @rx-angular/cdk
    - @rx-angular/template

- @rx-angular/state
- @rx-angular/isr

4. Blogbeitrag: https://www.netidee.at/rxangular/christmas-special (3.1.2025)

Abweichungen zum Plan: Verschiebung um 3 Monate nach hinten wegen Verlängerung von Arbeitspaketen 2-4.

4 Umsetzung Förderauflagen

Es wurden keine speziellen Förderauflagen festgelegt.

5 Liste Projektendergebnisse

1	Projektzwischenbericht	CC BY-SA 4.0	https://www.netidee.at/rxangular
2	Projektendbericht	CC BY-SA 4.0	https://www.netidee.at/rxangular
3	Entwickler_innen-DOKUMENTATION des Projektergebnisses für andere Entwickler_innen ("Dritte"), die das Projektergebnis nach Projektende nutzen/weiterentwickeln wollen	CC BY-SA 4.0	https://www.netidee.at/rxangular
4	Anwender_innen-DOKUMENTATION des Projektergebnisses für Anwender_innen, die das Projektergebnis nach Projektende nutzen wollen	CC BY-SA 4.0	https://www.netidee.at/rxangular
5	Veröffentlichungsfähiger Einseiter / Zusammenfassung	CC BY-SA 4.0	https://www.netidee.at/rxangular
6	Dokumentation Externkommunikation zur Erreichung Sichtbarkeit /Nachhaltigkeit (Teil des Endberichtes: Kapitel 11)	CC BY-SA 4.0	https://www.netidee.at/rxangular
7	SW-Projektergebnis-Teil_1: Modul zur Frame-Budget-Optimierung. Im Unterschied zum Stand der Technik ist es nicht erst nach Prozessende, sondern erstmals bereits zur Runtime möglich, das bereits verbrauchte Zeit-Budget eines Tasks zu ermitteln und so Maßnahmen zur Steuerung zu ergreifen (z.B. bei aufwendigem Listen-Rendering).	Open Source Lizenz: MIT	https://github.com/rx-angular/rx-angular https://www.netidee.at/rxangular

8	SW-Projektergebnis-Teil_2: Software-Modul für Task Abortion Mit dem bisherigen Stand der Technik bestand ein hohes Risiko, dass unnötige Arbeit ausgeführt wird, die für die Benutzenden nicht sichtbar ist (z.B. das Aktualisieren von Elementen, die im nächsten Task gelöscht werden). Dies wird mit unserer Lösung verhindert, indem derartige Tasks autom. erkannt und aus der Task Queue entfernt werden.	Open Source Lizenz: MIT	https://github.com/rx-angular/rx-angular https://www.netidee.at/rxangular
9	SW-Projektergebnis-Teil_3: Software-Modul für intelligentes Coalescing mit neuartigem Scoping-Mechanismus Eine zu rendernde Komponente wird mithilfe eines neu entwickelten Scoping-Algorithmus analysiert und mit einem Scoping-Flag (Markierung) versehen, damit sichergestellt wird, dass ausschließlich relevante Elemente für das nächste Render-Update herangezogen werden.	Open Source Lizenz: MIT	https://github.com/rx-angular/rx-angular https://www.netidee.at/rxangular

6 Verwertung der Projektergebnisse in der Praxis

Neben der Forschung und Entwicklungsarbeit bietet das Unternehmen umfassende Beratungen (z.B. Performance Consulting, Accessability Consulting), Workshops und Audits (Architecture Audits, Performance Audits, Accessability Audits) rund um das Thema Web Performance sowohl In-house als auch öffentlich an. Zentrales Thema ist stehts, wie mit limitierten System-Ressourcen und gegebenen Randbedingungen die bestmögliche Performance und der höchstmögliche Datendurchsatz erreicht werden kann.

7 Öffentlichkeitsarbeit/ Vernetzung

2024 fanden folgende Events mit Talks und Workshops des RxAngular-Teams statt:

10.02.2024: ngIndia, Delhi/India
10.02. Talk: 3 Reactive Primitives - Architecture in Modern Angular

24.02.2024: ngAsia
24.02. Performance at Scale: Non-trivial Optimization Techniques in Angular at NG Asia Meetup

18.03.–21.03.2024: ngConf, Salt Lake City/USA
18.03. Workshop: Nx for Scalable Angular Architectures (with Julian)
19.03. Workshop: Performance at Scale: High-Speed Angular Applications On Any Device (with Julian)
20.03. Talk: 3 Cornerstones ineffable in future Angular architecture

06.05.–08.05.2024: EnterJS, Mainz/Germany
06.05. Workshop: Modern Angular at Scale – Performance and Architecture
07.05. Talk: Modern Angular, Change Detection and Signals

08.05. Talk: How To Migrate Into a Monorepo: Strategies and Best Practices

24.05.2024: ngBelgrade, Belgrade/Serbia
24.05. Talk: Angular Performance and Core Web Vitals in 2024

25.06.–27.06.2024: ngRome, Rome/Italy
25.06. Workshop: High-speed Angular applications on any device
27.06. Talk: Angular Performance and Core Web Vitals in 2024

Monorepo World: 7 Oct 2024
07.10. The challenges of code quality and team KPIs in a monorepo (Michael Hladky)

ng-DE
09.10. High-speed Angular Apps on any device (Michael Hladky & Enea Jahollari)
10.10. Going deep into Signals Change Detection (Enea Jahollari)
11.10. Cut My Task Into Pieces - This is Concurrent Mode (Julian Jandl)

ngPoland: 6. November 2024 (Warsaw & Online)
Incremental Hydration in Angular (Michael Hladky)
Migrating to Modern Angular (Enea Jahollari)

JS Poland: 7. November 2024 (Warsaw & Online)
The Challenges of Code Quality and Team KPIs in a Monorepo (Michael Hladky)
Cut My Task Into Pieces - This is Concurrent Mode (Julian Jandl)

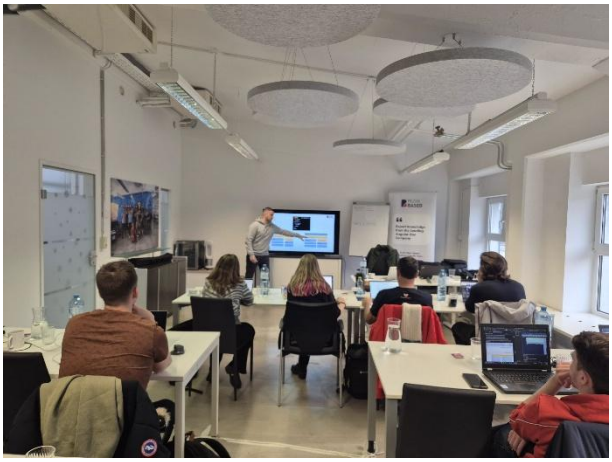
ng-BE: December 5 & 6, 2024
05.12. Architecting Angular Apps for High Performance: Control the Browser Render Pipeline, Master the Event Loop, Measure & Optimize CWV (Michael Hladky & Julian Jandl & Tanja Ulianova)
06.12. Incremental Hydration in Angular (Michael Hladky)

Unsere eigene Fachkonferenz: ngGlühwein (14.12.2024) - https://push-based.io/event/nggluehwein-2024
Mit Vorträgen u.a.:

- Migrating to Modern Angular (Enea Jahollari)
- Incremental Hydration in Angular (Michael Hladky)

In Workshops wurden u.a. folgende Themen behandelt:

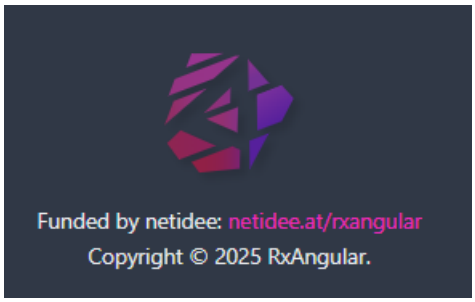
- The browser render pipeline
- How to analyze flame charts
- What are Core Web Vitals and how to measure them
- Best practices how to build high performance applications



8 Eigene Projektwebsite

https://www.rx-angular.io/

Auf die Finanzierung durch den Fördergeber wurde wie folgt hingewiesen:



9 Geplante Aktivitäten nach netidee-Projektende

Das Projekt wird fortgesetzt und Features werden kontinuierlich weiterentwickelt bzw. neu entwickelt. Das Feedback zu unserer ersten eigenen Angular-Fachkonferenz (ngGlühwein, 14.12.24) war überwiegend positiv, sodass wir diese Veranstaltung als jährliches Event planen.

Unsere Entwicklungen werden wir weiterhin auf Fachkonferenzen in Talks und Workshops präsentieren, u.a. auf der *NGVenice* in Venedig (13-14. März 2025).

NGVenice, Venice 13-14 March 2025

- Panel: Performance and SSR
  Speaker: Enea Jahollari
  Round-table about Performance and SSR with discussion starting from our speakers, but then open to interact with attendees who can ask questions on this topic.

- Incremental Hydration in Angular
  Speaker: Michael Hladky

Understand how the recent and future features play together and what the final puzzle could look like. We go deep on the past limitations of SSR and understand how event replay, hydration and deferred loading play together and unlock a powerful impact on your application

- Workshop: Building the Future with Angular: Master Signals and Deep Dive into Architecture Best Practices
  Trainers: Michael Hladky + Enea Jahollari

10 Anregungen für Weiterentwicklungen durch Dritte

Durch die Entwicklung als Open Source mit Copy-Left-Lizenz (MIT) ergeben sich umfassende Nutzungs- und Weiterentwicklungsmöglichkeiten. Jede:r kann unseren Code weiterentwickeln, implementieren, veröffentlichen und sogar Erlöse damit erzielen.

Wir haben im Förderzeitraum eine große API-Oberfläche entwickelt und weiterentwickelt, mit der unsere Software as-it-is verwendet werden kann. Mit dieser Entwicklungsschnittstelle kommen auch mannigfaltige Erweiterungsmöglichkeiten und Anleitungen zur Verwendung und Weiterentwicklung. Unteranderem haben wir im Förderzeitraum eine neue Funktion entwickelt, mit der Render-Strategien über unsere Strategievorlagen hinaus auf einfache Weise individualisiert werden können.

Dies ist auch unsere Weiterentwicklungsempfehlung: Renderstrategien sollen an die jeweilige Umgebung (z.B. in Embeded Systems mit besonderen Umgebungsmerkmalen) angepasst werden.

https://www.rx-angular.io/docs/state/api

Unsere Nutzungsempfehlung ist der Einsatz unseres Systems in performancekritischen Umgebungen, wie Public-facing Applikationen (z.B. Shop-Applikationen) oder Business Analytics.

11 Sichtbarkeit & Nachhaltigkeit

Sichtbarkeit

Unsere Maßnahmen zur Sichtbarkeit unseres Projekts waren Vorträge und Workshops bei 12 Fachkonferenzen und Meetups in 2024. Der Aufwand dafür ist groß, um umfasst u.a. das Vorbereiten der Vorträge, Vorbereiten von Demos vorbereiten und Reisezeit. Dennoch ist der Nutzen ebenso groß, da genau bei diesen Events unsere Zielgruppe präzise erreicht wird. Diese Konferenzen sind speziell für Entwickler:innen in unserer Nische ausgelegt und wir schätzen eine nahezu 100%ige Trefferquote.

Weitere Maßnahmen zur Sichtbarkeit sind regelmäßige Veröffentlichungen auf verschiedenen Social-Media-Kanälen wie LinkedIn oder Twitter/X. Besonders hervorgehoben werden kann hier Enea Jahollari mit über 10.000 Follower:innen . Dies ist in unserer Nische herausragend.

https://www.linkedin.com/company/push-based/
https://x.com/enea_jahollari

Lessons Learned

Unsere Lessons Learned und Empfehlungen für andere Projekte ist: Go Open Source! *Build in Public* ist ein vorteilhafter Entwicklungstreiber. Unsere Erfahrung ist, dass sich die Community unmittelbar und aktiv einschaltet. Man erhält sofort wertvolles Entwickler:innenfeedback.

Unser Beitrag zur Nachhaltigkeit

Wenn die Performance einer Webseite verbessert wird und somit weniger Arbeit am Client nötig ist, benötigt jeder Computer, der diese Webseite besucht, weniger Energie. Dies mag pro Client wenig erscheinen, in Summe jedoch ausschlaggebend. Z.B. weißt eine Webseite einer unserer Partner pro Tag 20 Mio. aktive Sessions auf. Durch den Einsatz unserer Tools benötigen somit 20 Mio. Clients weniger Energie.