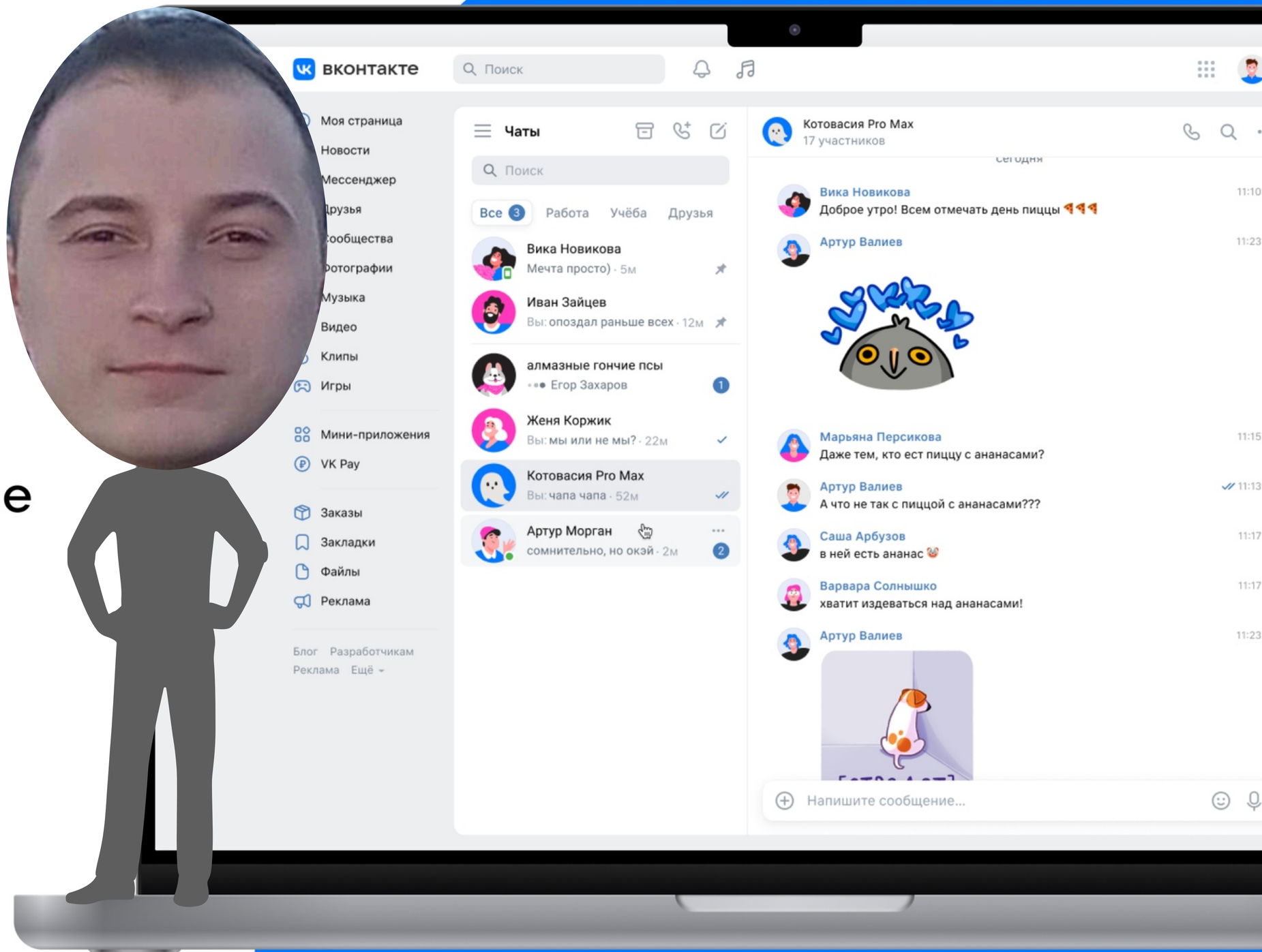
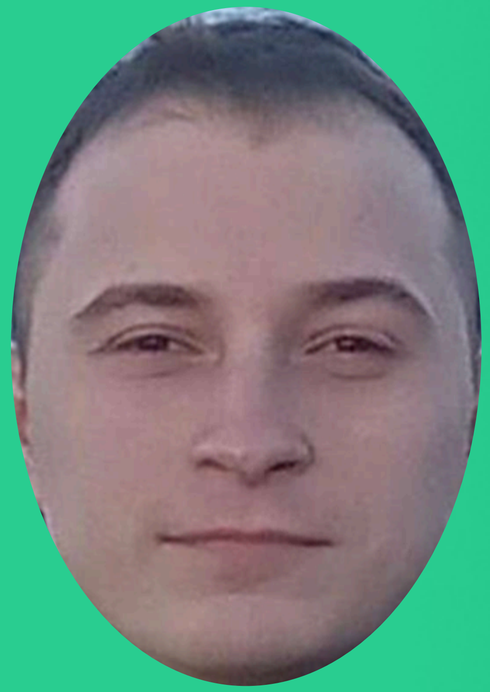
Папки с чатами

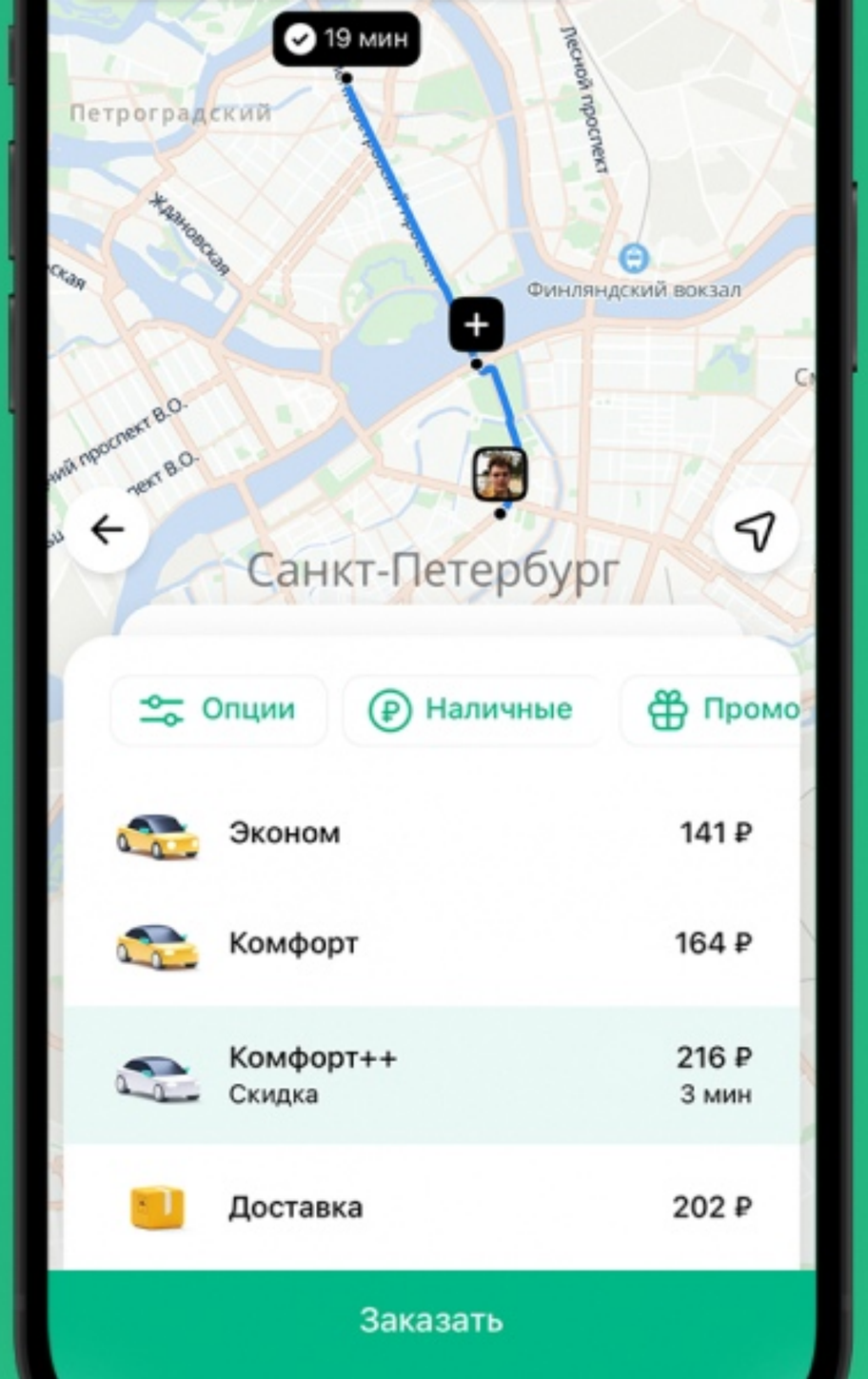
Меню действий
по правому клику

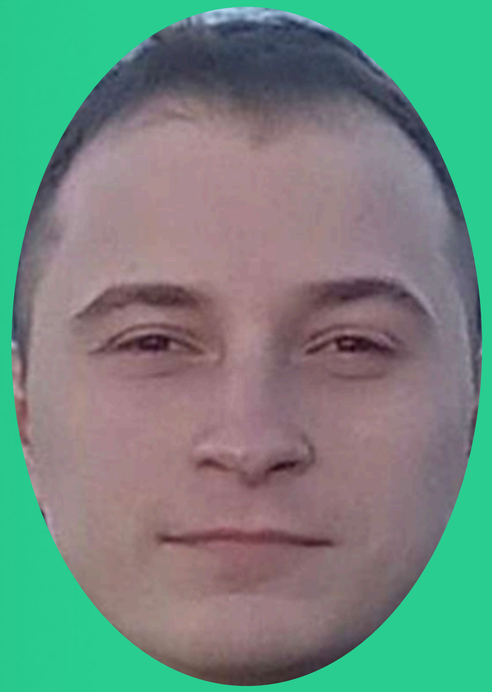
Тихие и исчезающие
сообщения

Анимация реакций






19 мин
Петроградский
Финляндский вокзал
Санкт-Петербург
Опции
Наличные
Промо
Эконом
141 ₽
Комфорт
164 ₽
Комфорт++
Скидка
216 ₽
3 мин
Доставка
202 ₽
Заказать




такси
вконтакте
Зенит
1925
575
такси
вконтакте
СБГ


vDOM
Runtime
Компилируемый
Signals
Bindings
Performance


vDOM
Runtime
Компилируемый
Signals
Bindings
Performance


vDOM
Runtime
Компилируемый
Signals
Bindings
Performance


vDOM
Runtime
Компилируемый
Signals
Bindings
Performance

VC.ru
https://vc.ru › dev › 708804-pervyi-vzglyad-na-solidjs

Первый взгляд на Solid.js

28 мая 2023 г. — За счет этого, Solid может работать напрямую с DOM, показывать значительно лучшую **производительность** и низкое потребление памяти. Пример ...

Время рендеринга при манипуляциях со строками таблицы:

Duration in milliseconds ± 95% confidence interval (Slowdown = Duration / Fastest)

Name Duration for...	vanillajs	solid-v1.5.4	solid-store-v1.5.4	react-hooks-v18.2.0	react-re-coil-v18.2.0 + 0.7.7	react-mobX-v17.0.1 + 5.15.4	react-re-dux-hooks-v18.2.0 + 8.0.5	react-zus-tand-v18.2.0 + 4.3.6
Implementation notes	772							
Implementation link	code	code	code	code	code	code	code	code
create rows creating 1,000 rows (5 warmup runs).	38.4 ±0.6 (1.00)	40.5 ±0.8 (1.05)	42.7 ±0.4 (1.11)	47.7 ±0.5 (1.24)	47.8 ±0.7 (1.24)	50.7 ±0.5 (1.32)	48.8 ±1.1 (1.27)	56.0 ±2.2 (1.46)
replace all rows updating all 1,000 rows (5 warmup runs).	42.2 ±0.7 (1.02)	43.4 ±0.7 (1.05)	46.1 ±0.8 (1.11)	53.1 ±0.8 (1.28)	53.1 ±0.6 (1.28)	52.4 ±0.4 (1.26)	54.4 ±0.7 (1.31)	56.8 ±0.8 (1.37)
partial update updating every 10th row for 1,000 rows (3 warmup runs). 16 x CPU slowdown.	97.9 ±2.2 (1.10)	99.1 ±2.5 (1.11)	95.8 ±2.6 (1.07)	119.8 ±2.9 (1.34)	121.8 ±2.5 (1.36)	114.0 ±2.2 (1.27)	129.0 ±3.0 (1.44)	136.3 ±2.5 (1.52)
select row highlighting a selected row. (5 warmup runs). 16 x CPU slowdown.	11.9 ±1.5 (1.24)	11.3 ±0.7 (1.18)	13.5 ±1.2 (1.42)	25.9 ±1.8 (2.72)	29.9 ±1.6 (3.14)	31.7 ±1.6 (3.32)	26.2 ±1.1 (2.75)	34.7 ±1.7 (3.64)
swap rows swap 2 rows for table with 1,000 rows. (5 warmup runs). 4 x CPU slowdown.	27.9 ±0.7 (1.04)	30.0 ±0.6 (1.12)	32.4 ±0.7 (1.21)	169.7 ±0.8 (6.32)	173.4 ±1.0 (6.46)	173.7 ±1.1 (6.47)	173.4 ±1.3 (6.46)	173.7 ±1.0 (6.47)
remove row removing one row. (5 warmup runs). 4 x CPU slowdown.	42.4 ±0.8 (1.01)	45.6 ±1.1 (1.08)	49.9 ±1.1 (1.19)	47.9 ±1.3 (1.14)	48.6 ±1.2 (1.16)	49.3 ±1.1 (1.17)	50.9 ±1.6 (1.21)	49.0 ±0.7 (1.17)
create many rows creating 10,000 rows. (5 warmup runs with 1k rows).	421.1 ±1.2 (1.00)	438.3 ±1.1 (1.04)	454.0 ±2.3 (1.08)	643.7 ±2.8 (1.53)	648.2 ±2.7 (1.54)	671.4 ±1.9 (1.60)	660.7 ±1.6 (1.57)	685.1 ±4.1 (1.63)
append rows to large table appending 1,000 to a table of 10,000 rows. 2 x CPU slowdown.	86.5 ±0.4 (1.00)	90.6 ±0.3 (1.05)	96.2 ±0.6 (1.11)	106.4 ±0.4 (1.23)	105.8 ±0.7 (1.22)	114.5 ±0.4 (1.32)	111.1 ±0.6 (1.28)	111.8 ±0.6 (1.29)

VC.ru
https://vc.r

Первый вз
28 мая 2023 г. ·
лучшую произ

Врем

Medium · Vladislav Lipatov
Отметок "Нравится": 3 · 2 года назад

С ReactJS на SolidJS - Vladislav Lipatov

SolidJS работает с нативными событиями браузера, однако для улучшения производительности SolidJS делегирует события, чтобы уменьшить количество ...

Давайте посмотрим на производительность.

Duration in millisec

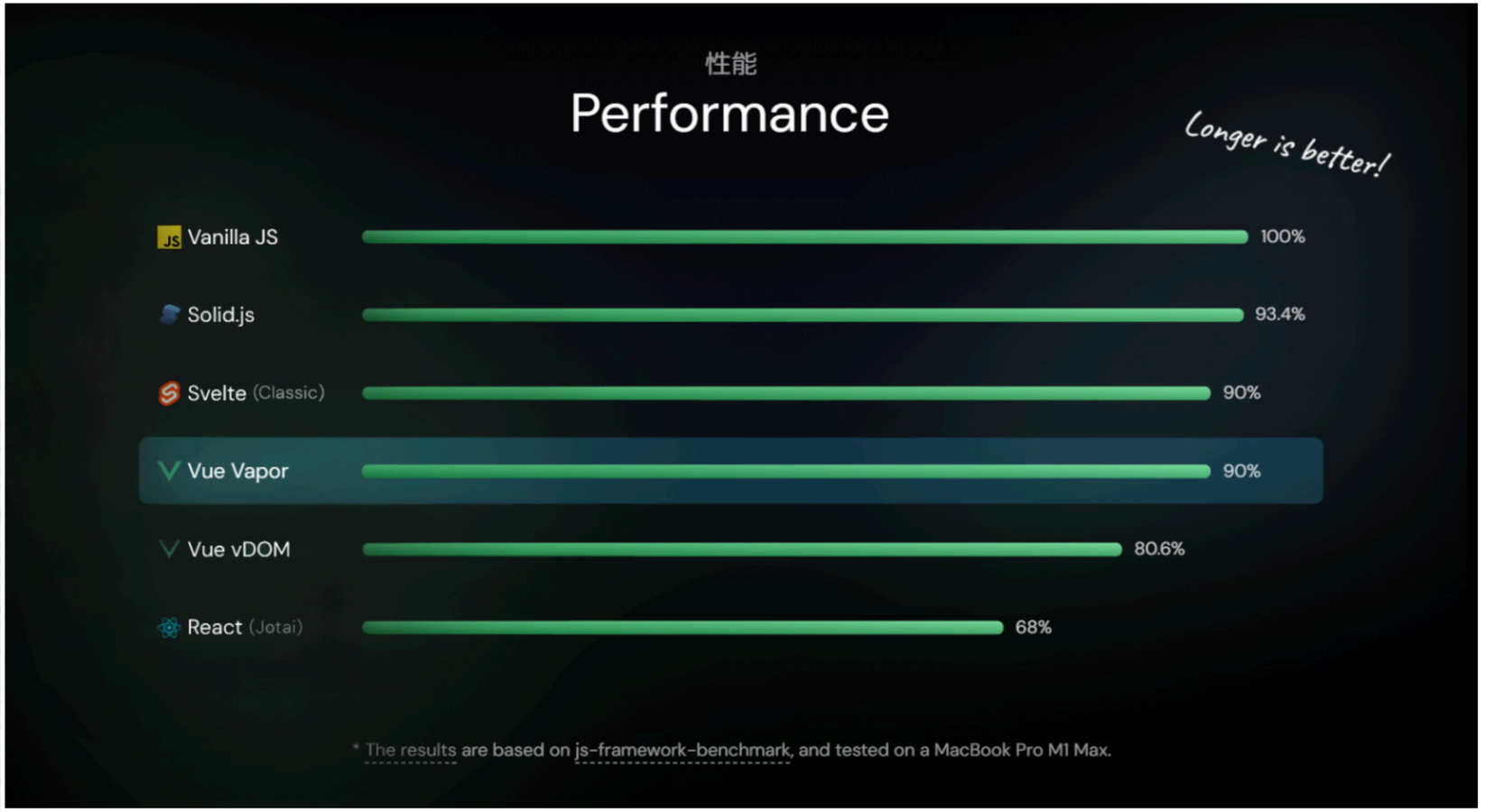
Name Duration for...	vanillajs	inferno-v7.4.8	solid-v1.5.4	lit-v2.6.1	vue-v3.2.47	svelte-v3.58.0	preact-v10.13.1	angular-v15.0.1	react-hooks-v18.2.0	react-mobX-v17.0.1 + 5.15.4	marko-v4.12.3	react-redux-hooks-immutable-v18.2.0 + 8.0.5	knockout-v3.5.0	ember-v4.10.0	react-redux-v18.2.0 + 8.0.5
Implementation notes	772			801									1139		
Implementation link	code	code	code	code	code	code	code	code	code	code	code	code	code	code	code
create rows creating 1,000 rows (5 warmup runs).	38.4 ±0.6 (1.00)	40.9 ±0.5 (1.06)	40.5 ±0.8 (1.05)	42.6 ±0.6 (1.11)	46.4 ±0.5 (1.21)	50.1 ±0.8 (1.30)	47.5 ±0.8 (1.24)	47.5 ±0.6 (1.24)	47.7 ±0.5 (1.24)	50.7 ±0.5 (1.32)	46.3 ±1.1 (1.21)	50.0 ±0.9 (1.30)	77.6 ±0.7 (2.02)	74.0 ±0.3 (1.93)	57.4 ±0.5 (1.49)
replace all rows updating all 1,000 rows (5 warmup runs).	42.2 ±0.7 (1.02)	43.3 ±0.5 (1.04)	43.4 ±0.7 (1.05)	47.3 ±0.6 (1.14)	47.5 ±0.7 (1.14)	53.7 ±0.4 (1.30)	58.7 ±0.7 (1.41)	50.6 ±0.7 (1.22)	53.1 ±0.8 (1.28)	52.4 ±0.4 (1.26)	49.5 ±0.6 (1.19)	57.7 ±1.1 (1.39)	89.0 ±0.4 (2.15)	92.6 ±0.8 (2.23)	64.4 ±0.8 (1.55)
partial update updating every 10th row for 1,000 rows (3 warmup runs). 16 x CPU slowdown.	97.9 ±2.2 (1.10)	100.6 ±2.5 (1.13)	99.1 ±2.5 (1.11)	109.6 ±3.0 (1.23)	118.4 ±2.3 (1.32)	106.4 ±2.7 (1.19)	123.8 ±2.8 (1.38)	104.4 ±2.6 (1.17)	119.8 ±2.9 (1.34)	114.0 ±2.2 (1.27)	142.1 ±1.8 (1.59)	146.6 ±2.1 (1.64)	105.0 ±1.5 (1.17)	142.3 ±3.4 (1.59)	148.9 ±4.0 (1.67)
select row highlighting a selected row. (5 warmup runs). 16 x CPU slowdown.	11.9 ±1.5 (1.24)	13.0 ±0.8 (1.36)	11.3 ±0.7 (1.18)	18.7 ±1.2 (1.96)	18.7 ±0.6 (1.96)	20.3 ±0.8 (2.13)	26.1 ±0.9 (2.74)	16.0 ±1.2 (1.68)	25.9 ±1.8 (2.72)	31.7 ±1.6 (3.32)	55.1 ±1.2 (5.78)	29.5 ±1.2 (3.10)	55.6 ±4.2 (5.83)	66.1 ±1.5 (6.93)	29.3 ±0.9 (3.08)
swap rows swap 2 rows for table with 1,000 rows. (5 warmup runs). 4 x CPU slowdown.	27.9 ±0.7 (1.04)	29.3 ±0.6 (1.09)	30.0 ±0.6 (1.12)	30.6 ±0.4 (1.14)	29.9 ±0.4 (1.11)	30.6 ±0.5 (1.14)	32.8 ±0.7 (1.22)	175.7 ±1.0 (6.54)	169.7 ±0.8 (6.32)	173.7 ±1.1 (6.47)	171.2 ±1.3 (6.38)	175.1 ±0.7 (6.52)	34.3 ±0.8 (1.28)	42.9 ±1.0 (1.60)	174.3 ±1.1 (6.49)
remove row removing one row. (5 warmup runs). 4 x CPU slowdown.	42.4 ±0.8 (1.01)	43.0 ±0.9 (1.02)	45.6 ±1.1 (1.08)	45.7 ±1.0 (1.09)	47.3 ±0.9 (1.13)	46.5 ±1.0 (1.11)	58.0 ±1.6 (1.38)	45.6 ±1.2 (1.08)	47.9 ±1.3 (1.14)	49.3 ±1.1 (1.17)	57.4 ±1.5 (1.37)	52.7 ±1.3 (1.25)	48.9 ±0.7 (1.16)	61.5 ±1.2 (1.46)	82.7 ±1.4 (1.97)
create many rows creating 10,000 rows. (5 warmup runs with 1k rows).	421.1 ±1.2 (1.00)	448.0 ±2.3 (1.07)	438.3 ±1.1 (1.04)	472.2 ±1.8 (1.13)	499.4 ±2.3 (1.19)	542.3 ±3.2 (1.29)	508.8 ±2.7 (1.21)	511.2 ±1.9 (1.22)	643.7 ±2.8 (1.53)	671.4 ±1.9 (1.60)	493.4 ±1.4 (1.18)	668.6 ±2.4 (1.59)	768.1 ±8.6 (1.83)	697.4 ±2.6 (1.66)	748.5 ±2.4 (1.78)
append rows to large table appending 1,000 to a table of 10,000 rows. 2 x CPU slowdown.	86.5 ±0.4 (1.00)	92.3 ±0.7 (1.07)	90.6 ±0.3 (1.05)	97.8 ±0.4 (1.13)	99.4 ±0.7 (1.15)	114.7 ±0.7 (1.33)	111.6 ±0.6 (1.29)	106.2 ±0.4 (1.23)	106.4 ±0.4 (1.23)	114.5 ±0.4 (1.32)	104.7 ±0.8 (1.21)	116.2 ±1.1 (1.34)	161.2 ±1.7 (1.86)	176.5 ±1.2 (2.04)	129.8 ±0.7 (1.50)
clear rows clearing a table with 1,000 rows. 8 x CPU slowdown. (5 warmup runs).	31.3 ±0.5 (1.04)	32.8 ±0.7 (1.09)	38.3 ±0.4 (1.27)	41.3 ±1.1 (1.37)	39.5 ±0.8 (1.31)	41.3 ±0.7 (1.37)	41.6 ±0.9 (1.38)	68.7 ±1.3 (2.28)	58.4 ±1.0 (1.94)	51.0 ±1.6 (1.69)	52.2 ±1.2 (1.73)	62.8 ±0.6 (2.08)	107.6 ±1.0 (3.57)	75.3 ±1.0 (2.50)	78.3 ±1.3 (2.60)
geometric mean of all factors in the table	1.05	1.10	1.10	1.23	1.26	1.33	1.42	1.61	1.75	1.79	1.87	1.91	2.01	2.14	2.17

Хабр

https://habr.com › companies › first › articles

Новый этап эволюции Vue — Vapor

7 нояб. 2024 г. — Подводя итоги · **Vapor — это новое начало для Vue**. Было переписано почти всё с нуля. · Vapor является подмножеством режима vDOM Vue, с акцентом ...



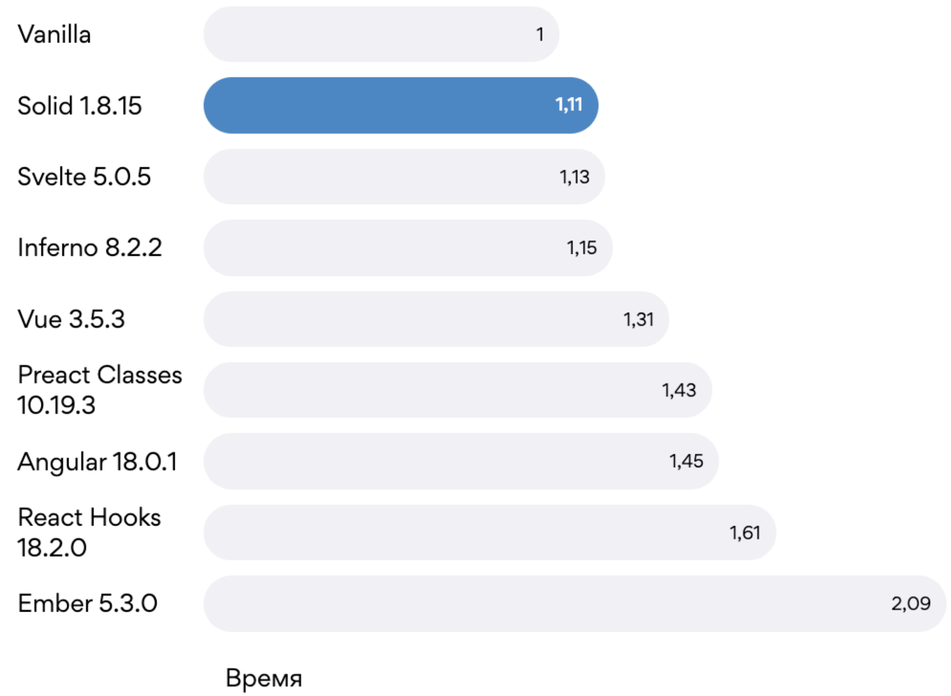
Результат получен с помощью js-framework-benchmark, и протестировано на MacBook Pro M1 Max

SVELTE

【docs】Suggest Svelte provide official benchmark in page loading and memory usage #7522

https://github.com/krausest/js-framework-benchmark is pretty conclusive and it's one less thing for use to maintain. I don't think it's worth duplicating the benchmark, plus it can be biased too.

While there's tradeoffs with the approach, I think it's not consistent with the culture of the maintainers community to directly compare to other frameworks and so a benchmark is something unlikely to be hosted on the official site.



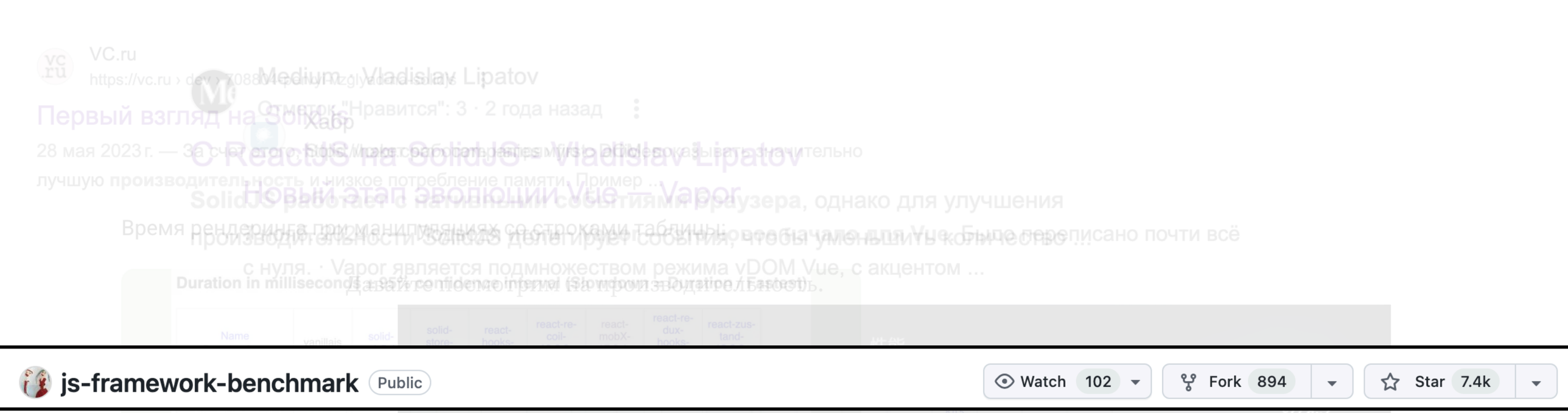
Показать больше клиентских и серверных бенчмарков >

Ориентирован на производительность

как на клиенте, так и на сервере

Сила мелкозернистой реактивности как подхода сияет во всех заметных бенчмарках. Даже если производительность не является Вашей целью, то она достигается практически без усилий и опыта разработки. Воспринимайте прирост производительности Solid как халявную победу. Solid - это возможность быть быстрым без усилий

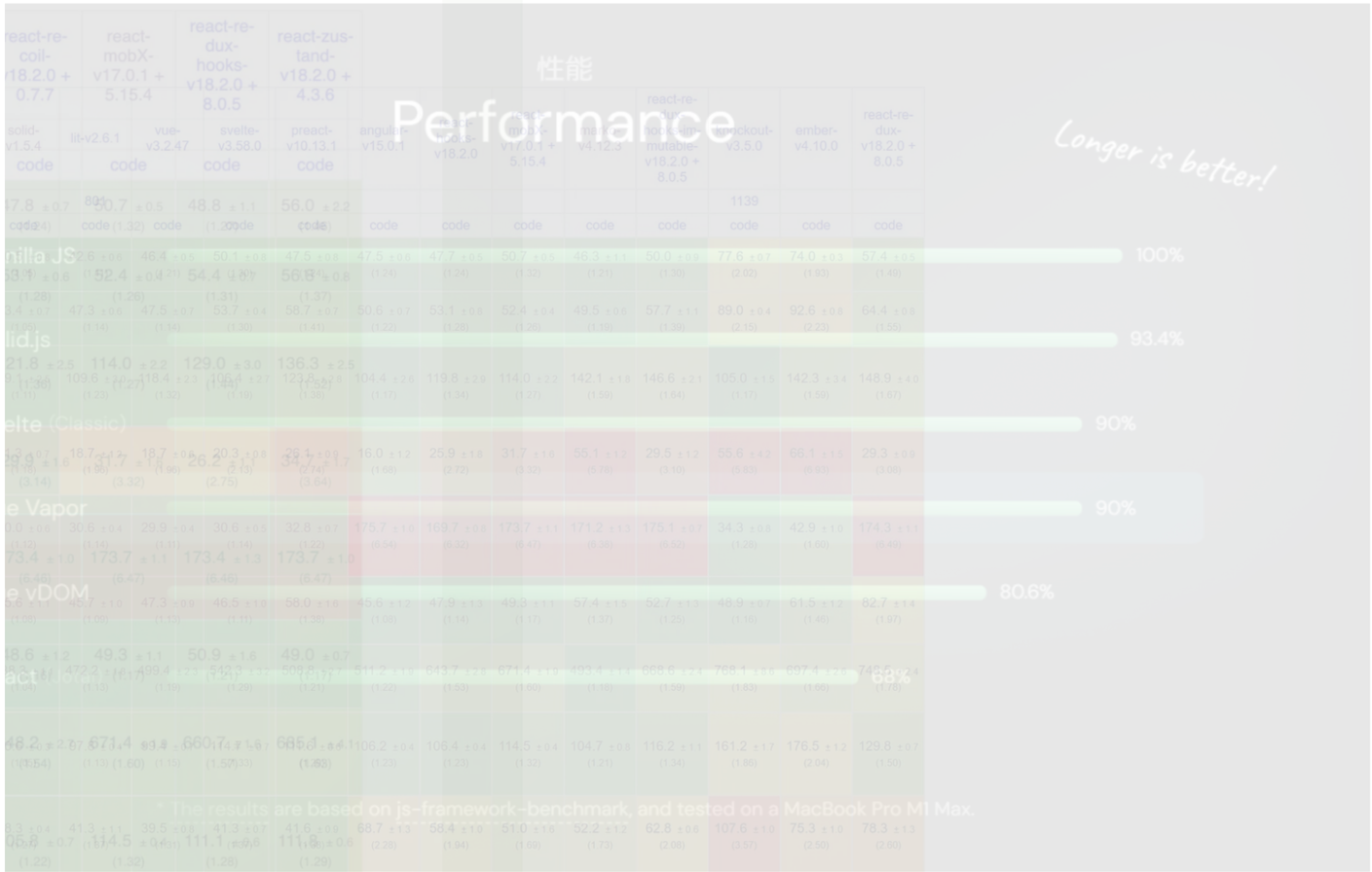
Читать полную статью >



Наша подборка

Name Duration for...	solid-v1.9.3	svelte-v5.42.1	react-hooks-v19.2.0	react-mobX-v19.0.0 + 6.13.5
Implementation notes				
Implementation link	code	code	code	code
create rows creating 1,000 rows. (5 warmup runs).	90.2 ±1.1 (1.00)	92.8 ±1.0 (1.03)	109.7 ±1.0 (1.22)	118.6 ±1.1 (1.31)
replace all rows updating all 1,000 rows. (5 warmup runs).	97.9 ±1.2 (1.00)	99.4 ±0.9 (1.02)	121.2 ±1.0 (1.24)	130.7 ±2.0 (1.34)
partial update updating every 10th row for 1,000 row. (3 warmup runs). 4 x CPU slowdown.	55.6 ±0.4 (1.00)	56.4 ±0.9 (1.01)	68.6 ±0.7 (1.23)	72.5 ±0.9 (1.30)
select row highlighting a selected row. (5 warmup runs). 4 x CPU slowdown.	14.5 ±0.7 (1.00)	17.9 ±1.0 (1.23)	18.2 ±0.4 (1.26)	20.1 ±0.8 (1.39)
swap rows swap 2 rows for table with 1,000 rows. (5 warmup runs). 4 x CPU slowdown.	65.4 ±0.9 (1.01)	64.8 ±1.4 (1.00)	370.0 ±5.0 (5.71)	353.0 ±3.3 (5.45)
remove row removing one row. (5 warmup runs). 2 x CPU slowdown.	47.9 ±1.1 (1.01)	47.3 ±1.1 (1.00)	53.6 ±0.8 (1.13)	54.4 ±0.7 (1.15)
create many rows creating 10,000 rows. (5 warmup runs).	937.4 ±5.2 (1.00)	933.3 ±8.4 (1.00)	1,314.0 ±9.0 (1.41)	1,403.7 ±11.4 (1.50)
append rows to large table appending 1,000 to a table of 1,000 rows. (5 warmup runs).	110.6 ±1.2 (1.00)	110.8 ±1.4 (1.00)	123.4 ±0.9 (1.12)	132.6 ±1.1 (1.20)
clear rows clearing a table with 1,000 rows. (5 warmup runs). 4 x CPU slowdown.	39.9 ±0.9 (1.14)	35.0 ±0.7 (1.00)	60.9 ±0.8 (1.74)	65.0 ±0.7 (1.86)
weighted geometric mean of all factors in the table	1.02	1.02	1.33	1.41

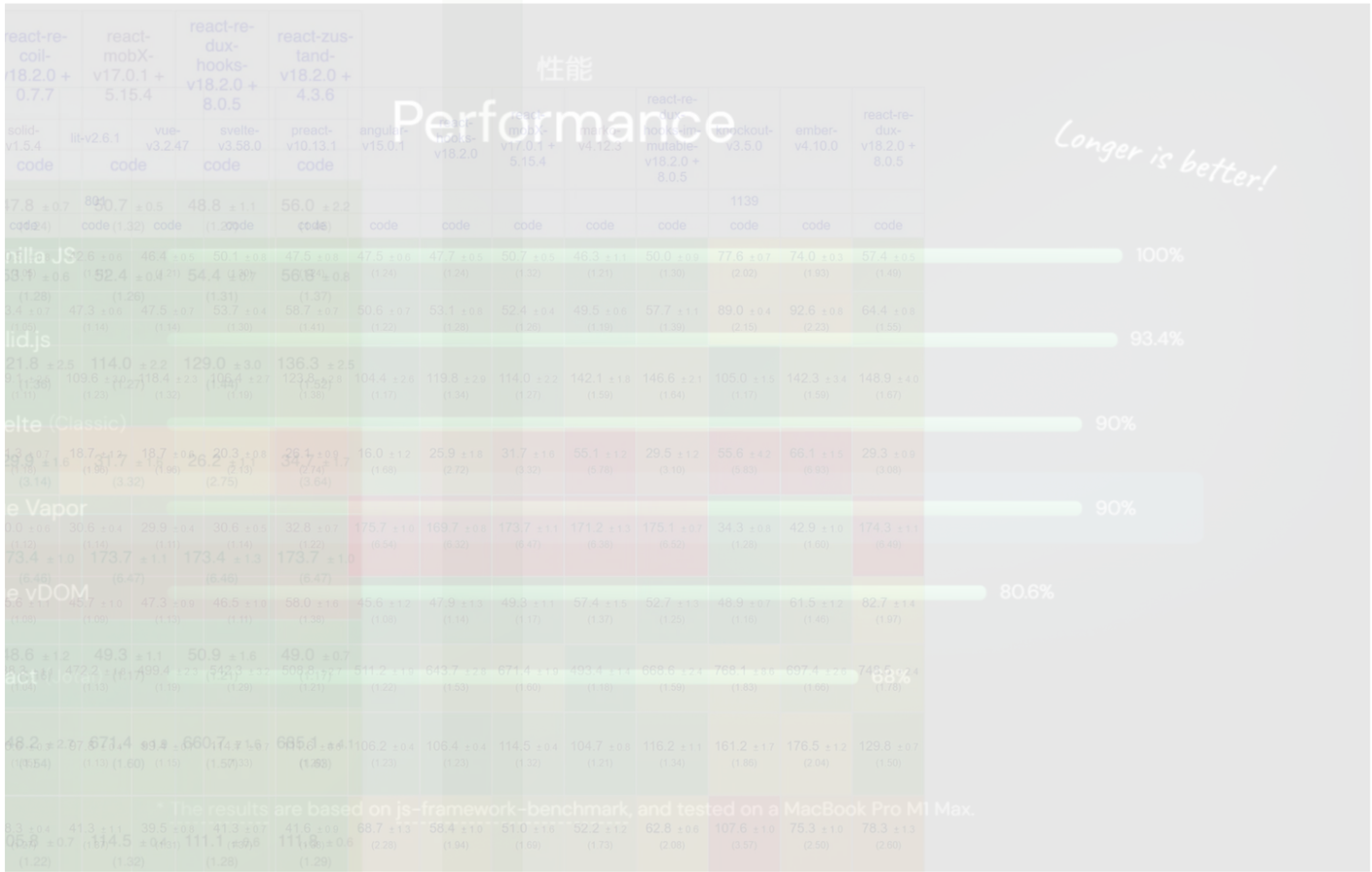
Some might recognise the concept of "Signals" in the concepts described below. This is correct, MobX is a signal based state management library avant la lettre.



Наша подборка

Name Duration for...	solid-v1.9.3	svelte-v5.42.1	react-hooks-v19.2.0	react-mobX-v19.0.0 + 6.13.5
Implementation notes				
Implementation link	code	code	code	code
create rows creating 1,000 rows. (5 warmup runs).	90.2 ±1.1 (1.00)	92.8 ±1.0 (1.03)	109.7 ±1.0 (1.22)	118.6 ±1.1 (1.31)
replace all rows updating all 1,000 rows. (5 warmup runs).	97.9 ±1.2 (1.00)	99.4 ±0.9 (1.02)	121.2 ±1.0 (1.24)	130.7 ±2.0 (1.34)
partial update updating every 10th row for 1,000 row. (3 warmup runs). 4 x CPU slowdown.	55.6 ±0.4 (1.00)	56.4 ±0.9 (1.01)	68.6 ±0.7 (1.23)	72.5 ±0.9 (1.30)
select row highlighting a selected row. (5 warmup runs). 4 x CPU slowdown.	14.5 ±0.7 (1.00)	17.9 ±1.0 (1.23)	18.2 ±0.4 (1.26)	20.1 ±0.8 (1.39)
swap rows swap 2 rows for table with 1,000 rows. (5 warmup runs). 4 x CPU slowdown.	65.4 ±0.9 (1.01)	64.8 ±1.4 (1.00)	370.0 ±5.0 (5.71)	353.0 ±3.3 (5.45)
remove row removing one row. (5 warmup runs). 2 x CPU slowdown.	47.9 ±1.1 (1.01)	47.3 ±1.1 (1.00)	53.6 ±0.8 (1.13)	54.4 ±0.7 (1.15)
create many rows creating 10,000 rows. (5 warmup runs).	937.4 ±5.2 (1.00)	933.3 ±8.4 (1.00)	1,314.0 ±9.0 (1.41)	1,403.7 ±11.4 (1.50)
append rows to large table appending 1,000 to a table of 1,000 rows. (5 warmup runs).	110.6 ±1.2 (1.00)	110.8 ±1.4 (1.00)	123.4 ±0.9 (1.12)	132.6 ±1.1 (1.20)
clear rows clearing a table with 1,000 rows. (5 warmup runs). 4 x CPU slowdown.	39.9 ±0.9 (1.14)	35.0 ±0.7 (1.00)	60.9 ±0.8 (1.74)	65.0 ±0.7 (1.86)
weighted geometric mean of all factors in the table	1.02	1.02	1.33	1.41

Some might recognise the concept of "Signals" in the concepts described below. This is correct, MobX is a signal based state management library avant la lettre.

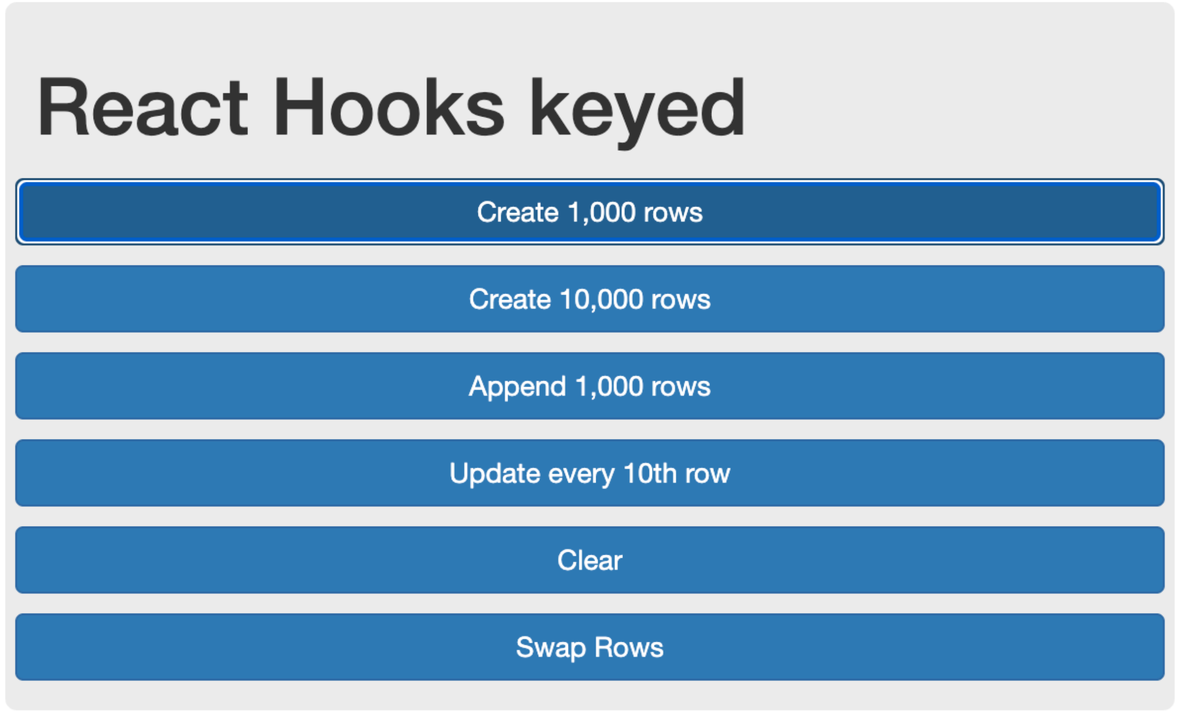


Наша подборка

Name Duration for...	solid-v1.9.3	svelte-v5.42.1	react-hooks-v19.2.0	react-mobX-v19.0.0 + 6.13.5
Implementation notes				
Implementation link	code	code	code	code
create rows creating 1,000 rows. (5 warmup runs).	90.2 ±1.1 (1.00)	92.8 ±1.0 (1.03)	109.7 ±1.0 (1.22)	118.6 ±1.1 (1.31)
replace all rows updating all 1,000 rows. (5 warmup runs).	97.9 ±1.2 (1.00)	99.4 ±0.9 (1.02)	121.2 ±1.0 (1.24)	130.7 ±2.0 (1.34)
partial update updating every 10th row for 1,000 row. (3 warmup runs). 4 x CPU slowdown.	55.6 ±0.4 (1.00)	56.4 ±0.9 (1.01)	68.6 ±0.7 (1.23)	72.5 ±0.9 (1.30)
select row highlighting a selected row. (5 warmup runs). 4 x CPU slowdown.	14.5 ±0.7 (1.00)	17.9 ±1.0 (1.23)	18.2 ±0.4 (1.26)	20.1 ±0.8 (1.39)
swap rows swap 2 rows for table with 1,000 rows. (5 warmup runs). 4 x CPU slowdown.	65.4 ±0.9 (1.01)	64.8 ±1.4 (1.00)	370.0 ±5.0 (5.71)	353.0 ±3.3 (5.45)
remove row removing one row. (5 warmup runs). 2 x CPU slowdown.	47.9 ±1.1 (1.01)	47.3 ±1.1 (1.00)	53.6 ±0.8 (1.13)	54.4 ±0.7 (1.15)
create many rows creating 10,000 rows. (5 warmup runs).	937.4 ±5.2 (1.00)	933.3 ±8.4 (1.00)	1,314.0 ± 9.0 (1.41)	1,403.7 ± 11.4 (1.50)
append rows to large table appending 1,000 to a table of 1,000 rows. (5 warmup runs).	110.6 ±1.2 (1.00)	110.8 ±1.4 (1.00)	123.4 ±0.9 (1.12)	132.6 ±1.1 (1.20)
clear rows clearing a table with 1,000 rows. (5 warmup runs). 4 x CPU slowdown.	39.9 ±0.9 (1.14)	35.0 ±0.7 (1.00)	60.9 ±0.8 (1.74)	65.0 ±0.7 (1.86)
weighted geometric mean of all factors in the table	1.02	1.02	1.33	1.41

Some might recognise the concept of "Signals" in the concepts described below. This is correct, MobX is a signal based state management library avant la lettre.

Create (many) rows



Create (many) rows

Name Duration for...	solid-v1.9.3	svelte-v5.42.1	react-hooks-v19.2.0	react-mobX-v19.0.0 + 6.13.5
Implementation notes				
Implementation link	code	code	code	code
create rows creating 1,000 rows. (5 warmup runs).	90.2 ± 1.1 (1.00)	92.8 ± 1.0 (1.03)	109.7 ± 1.0 (1.22)	118.6 ± 1.1 (1.31)
replace all rows				
slowdown:				
create many rows creating 10,000 rows. (5 warmup runs).	937.4 ± 5.2 (1.00)	933.3 ± 8.4 (1.00)	1,314.0 ± 9.0 (1.41)	1,403.7 ± 11.4 (1.50)

Create (many) rows

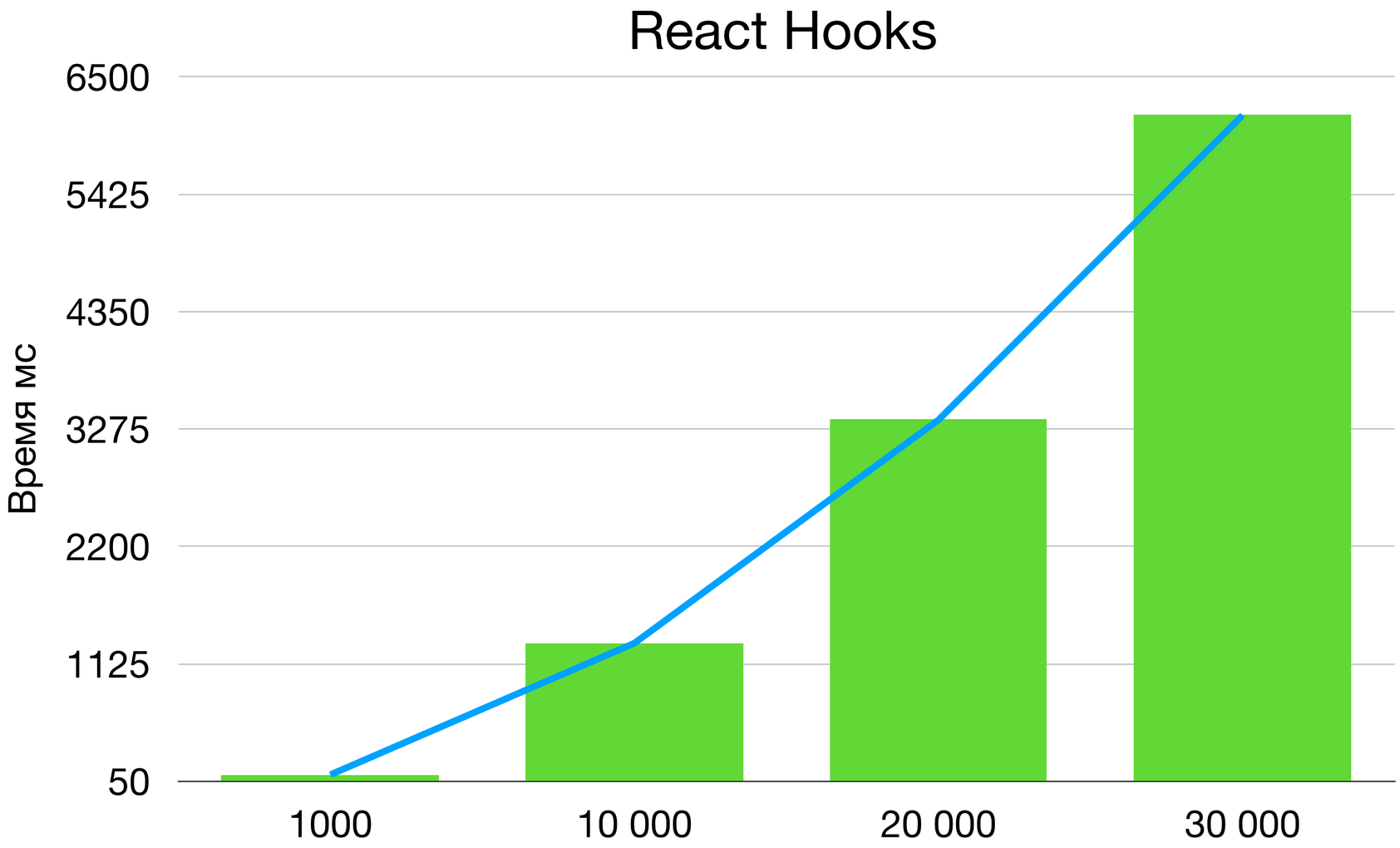
Name Duration for...	solid-v1.9.3	svelte-v5.42.1	react-hooks-v19.2.0	react-mobX-v19.0.0 + 6.13.5
Implementation notes				
Implementation link	code	code	code	code
create rows creating 1,000 rows. (5 warmup runs).	90.2 ± 1.1 (1.00)	92.8 ± 1.0 (1.03)	109.7 ± 1.0 (1.22)	118.6 ± 1.1 (1.31)
create many rows creating 10,000 rows. (5 warmup runs).	937.4 ± 5.2 (1.00)	933.3 ± 8.4 (1.00)	1,314.0 ± 9.0 (1.41)	1,403.7 ± 11.4 (1.50)
Отношение времени затраченного на отрисовку 10 000 строк ко времени на отрисовку 1000 строк	~10	~10	~12	~12

Create (many) rows

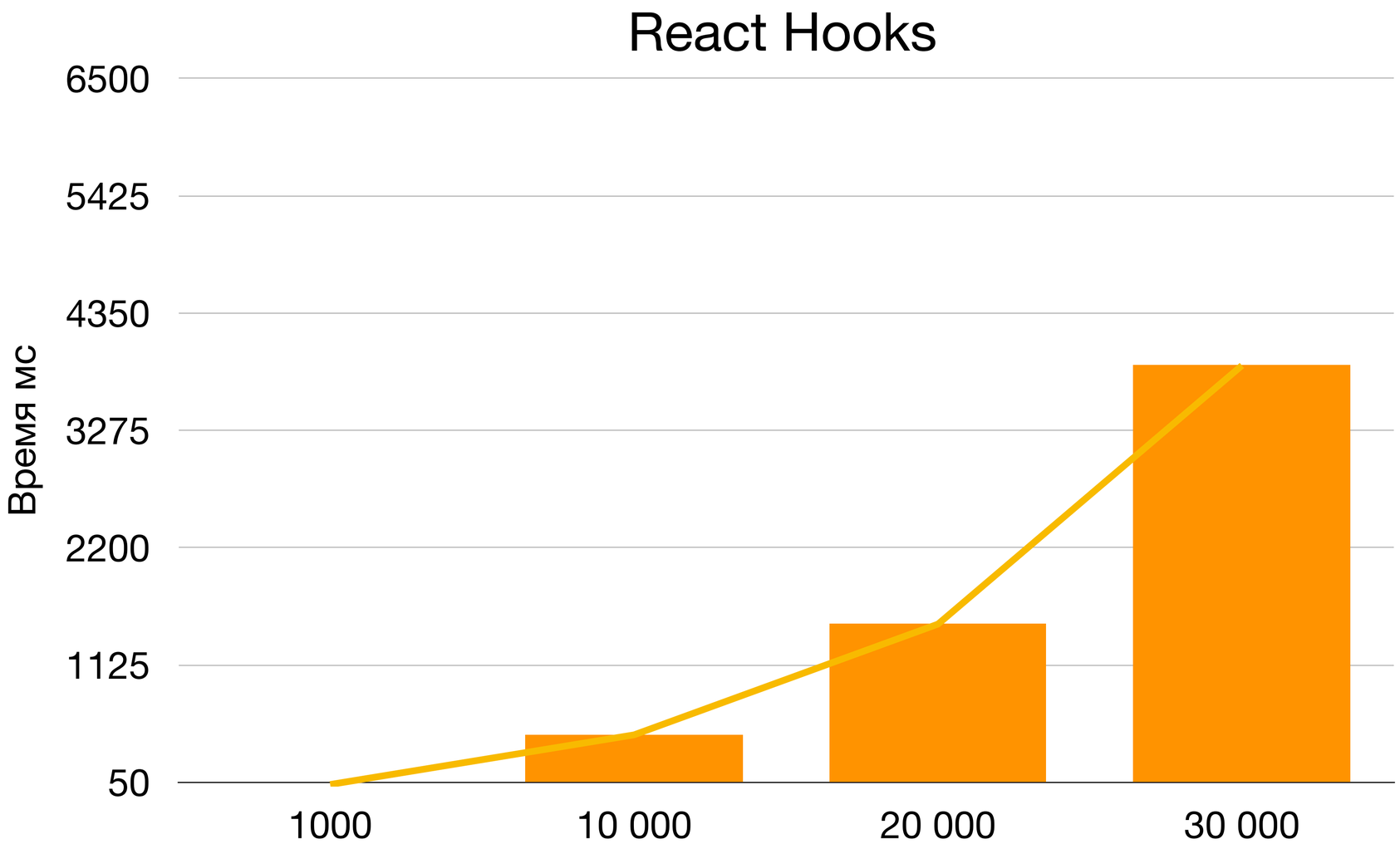
Name Duration for...	solid-v1.9.3	svelte-v5.42.1	react-hooks-v19.2.0	react-mobX-v19.0.0 + 6.13.5
Implementation notes				
Implementation link	code	code	code	code
create rows creating 1,000 rows. (5 warmup runs).	90.2 ± 1.1 (1.00)	92.8 ± 1.0 (1.03)	109.7 ± 1.0 (1.22)	118.6 ± 1.1 (1.31)
create many rows creating 10,000 rows. (5 warmup runs).	937.4 ± 5.2 (1.00)	933.3 ± 8.4 (1.00)	1,314.0 ± 9.0 (1.41)	1,403.7 ± 11.4 (1.50)
Отношение времени затраченного на отрисовку 10 000 строк ко времени на отрисовку 1000 строк	~10	~10	~12	~12

> 200+ ms

Create (many) rows



Create (many) rows



Что внутри?

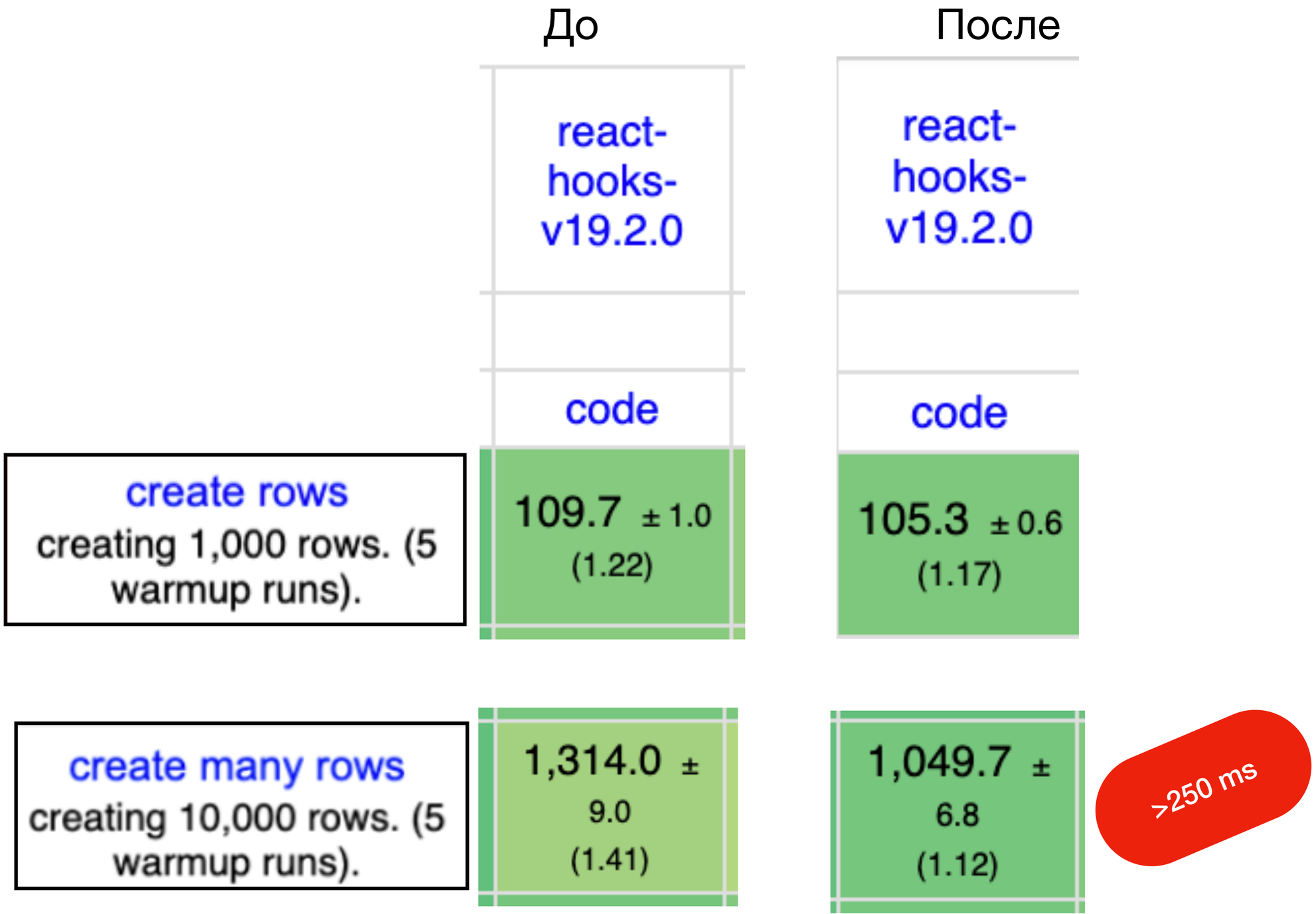
```
const Main = () => {
  const [{ data, selected }, dispatch] = useReducer(listReducer, initialState);

  return (<div className="container">
    <Jumbotron dispatch={dispatch} />
    <table className="table table-hover table-striped test-data">
      <tbody>
        {data.map(item => (
          <Row key={item.id} item={item} selected={selected === item.id} dispatch={dispatch} />
        ))}
      </tbody>
    </table>
    <span className="preloadicon glyphicon glyphicon-remove" aria-hidden="true" />
  </div>);
}
```

Что если ...

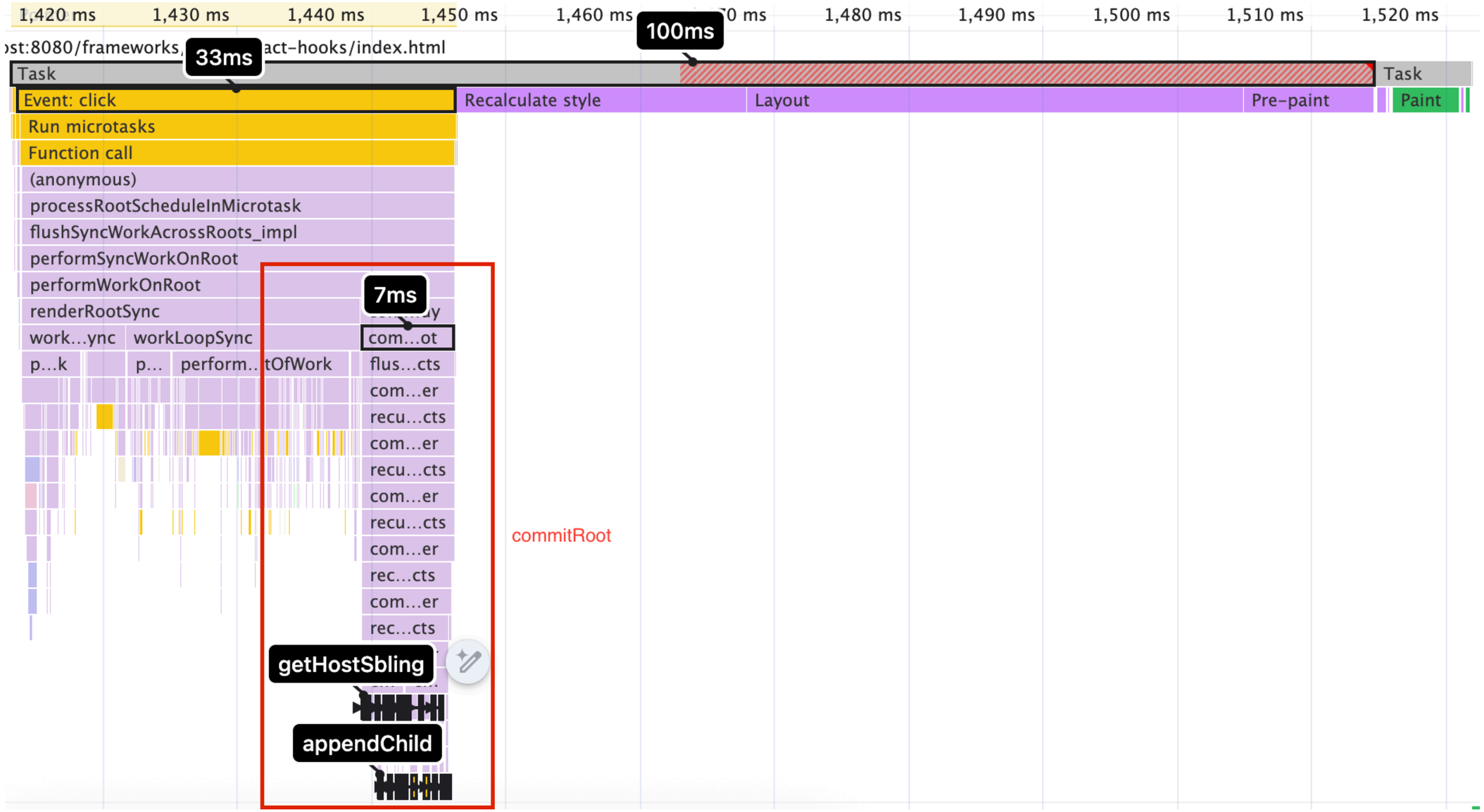
```
const Main = () => {
  const [{ data, selected }, dispatch] = useReducer(listReducer, initialState);

  return (<div className="container">
    <Jumbotron dispatch={dispatch} />
    <table className="table table-hover table-striped test-data">
      {!!data.length &&      You, 1 second ago • Uncommitted changes
        <tbody>
        {data.map(item => (
          <Row key={item.id} item={item} selected={selected === item.id} dispatch={dispatch} />
        ))}
        </tbody>
      }
    </table>
    <span className="preloadicon glyphicon glyphicon-remove" aria-hidden="true" />
  </div>);
}
```

Create (many) rows

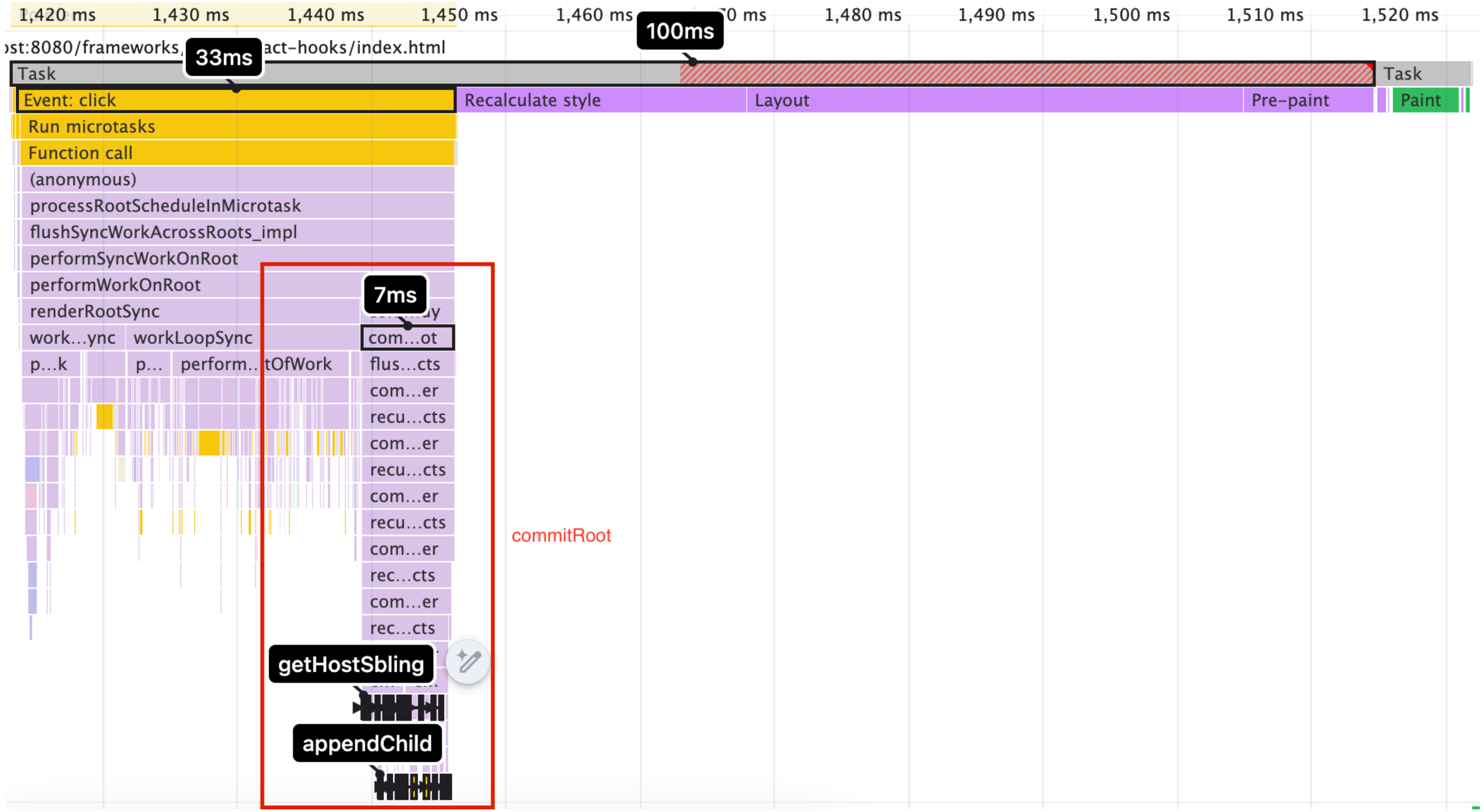
Name Duration for...	solid-v1.9.3	svelte-v5.42.1	react-hooks-v19.2.0	react-mobX-v19.0.0 + 6.13.5
Implementation notes				
Implementation link	code	code	code	code
create rows creating 1,000 rows. (5 warmup runs).	90.2 ± 1.1 (1.00)	92.8 ± 1.0 (1.03)	105.3 ± 0.6 (1.17)	114.0 ± 1.0 (1.26)
create many rows creating 10,000 rows. (5 warmup runs).	937.4 ± 5.2 (1.00)	933.3 ± 8.4 (1.00)	1,049.7 ± 6.8 (1.12)	1,117.3 ± 9.1 (1.20)
Отношение времени затраченного на отрисовку 10 000 строк ко времени на отрисовку 1000 строк	~10	~10	~10	~10


До
После
react-hooks-v19.2.0
react-hooks-v19.2.0
code
code
create rows
creating 1,000 rows. (5 warmup runs).
109.7 ± 1.0
(1.22)
105.3 ± 0.6
(1.17)
create many rows
creating 10,000 rows. (5 warmup runs).
1,314.0 ± 9.0
(1.41)
1,049.7 ± 6.8
(1.12)
>250 ms

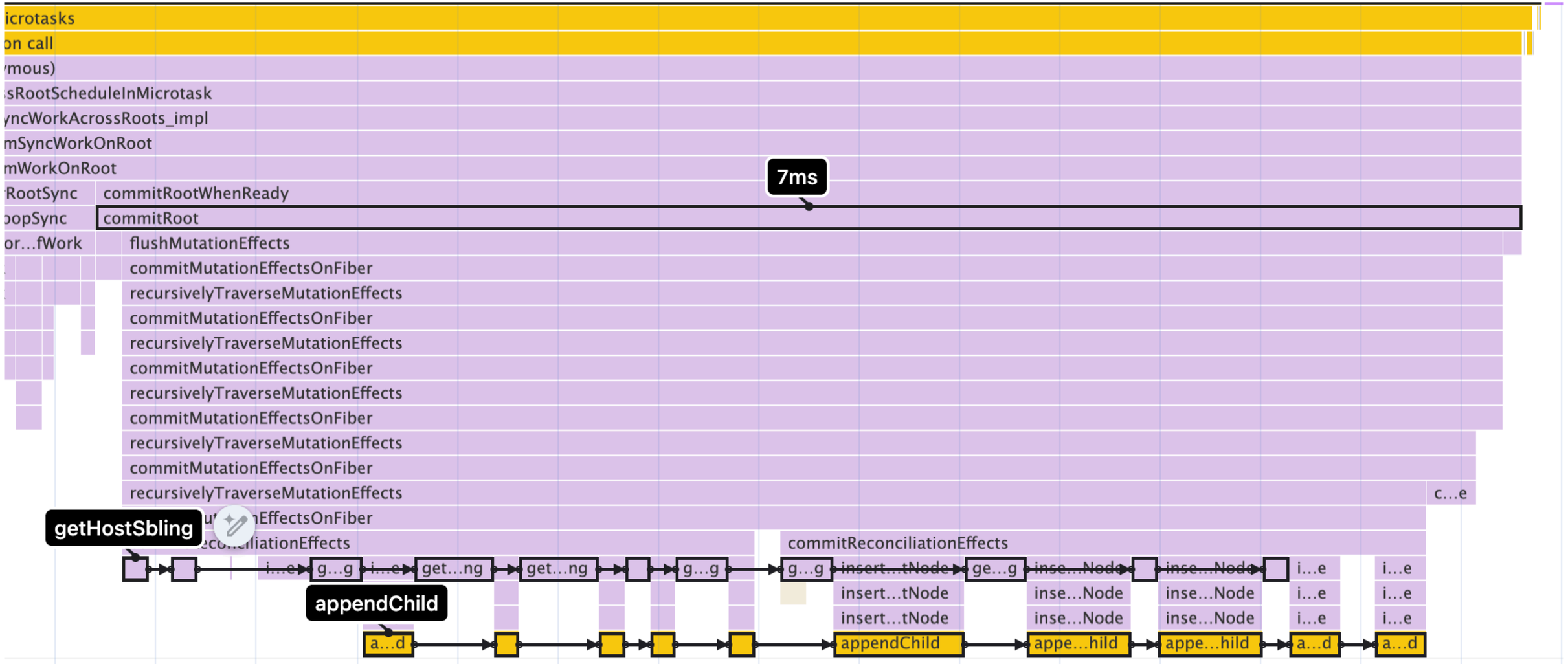
Create 1000 rows



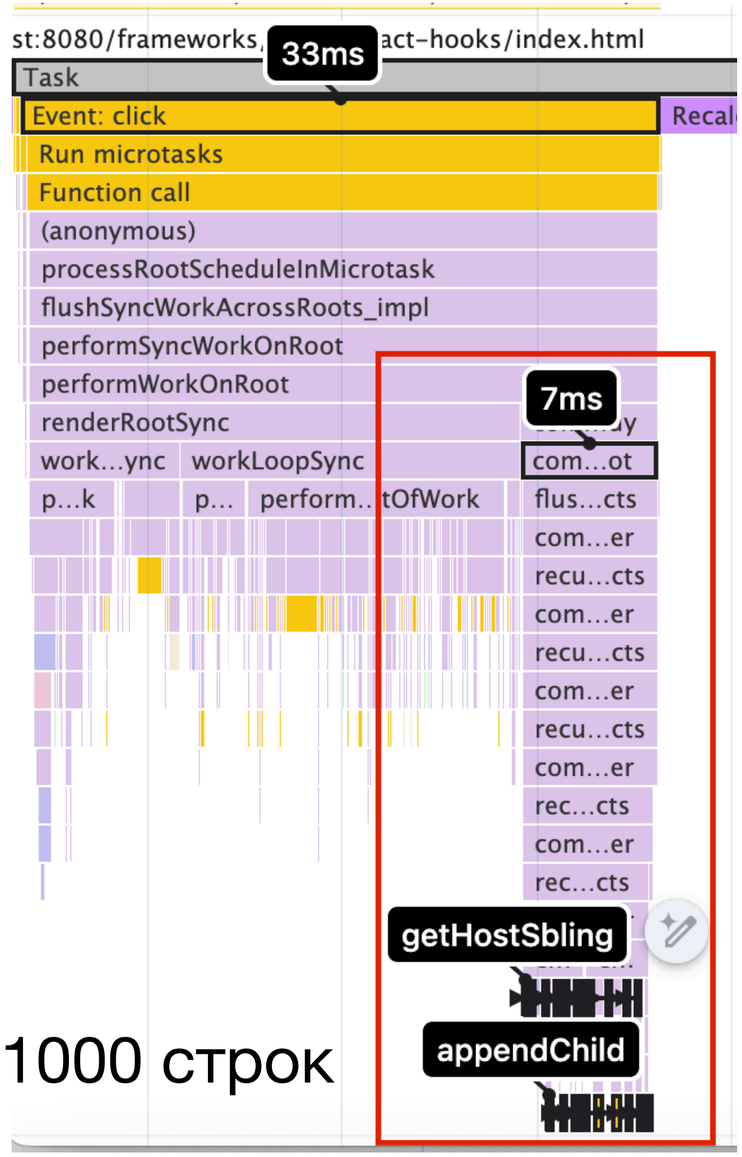
Create 1000 rows



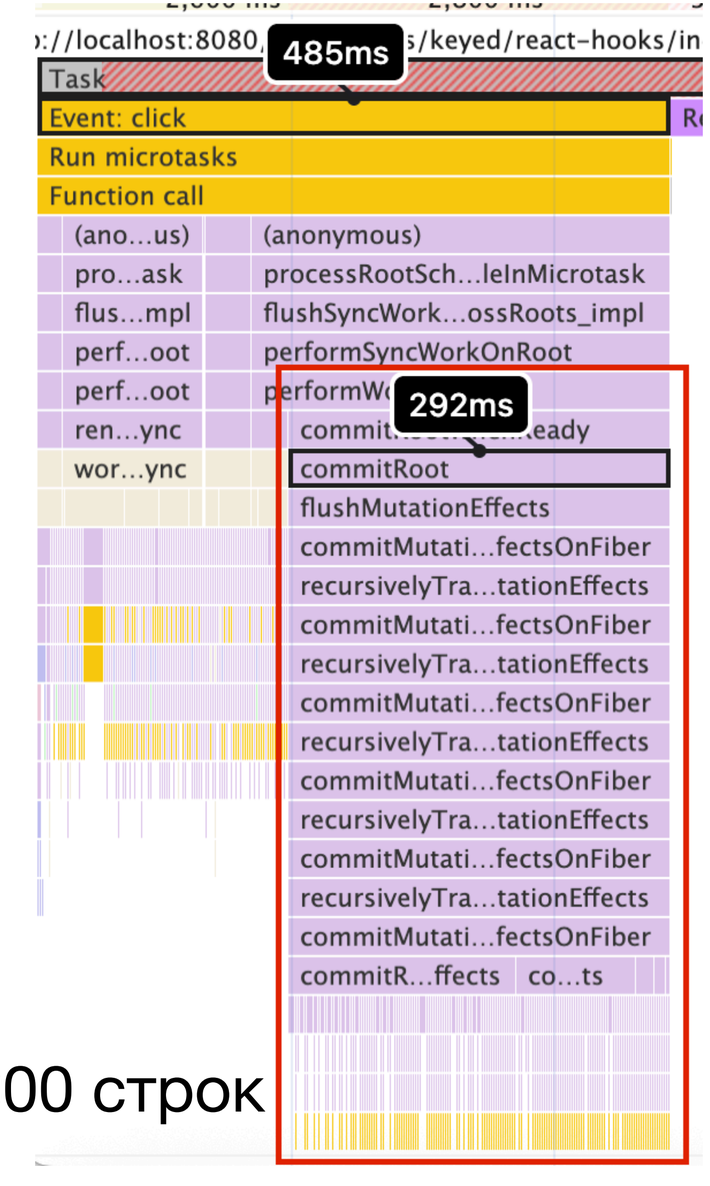
Create 1000 rows



Фаза коммит

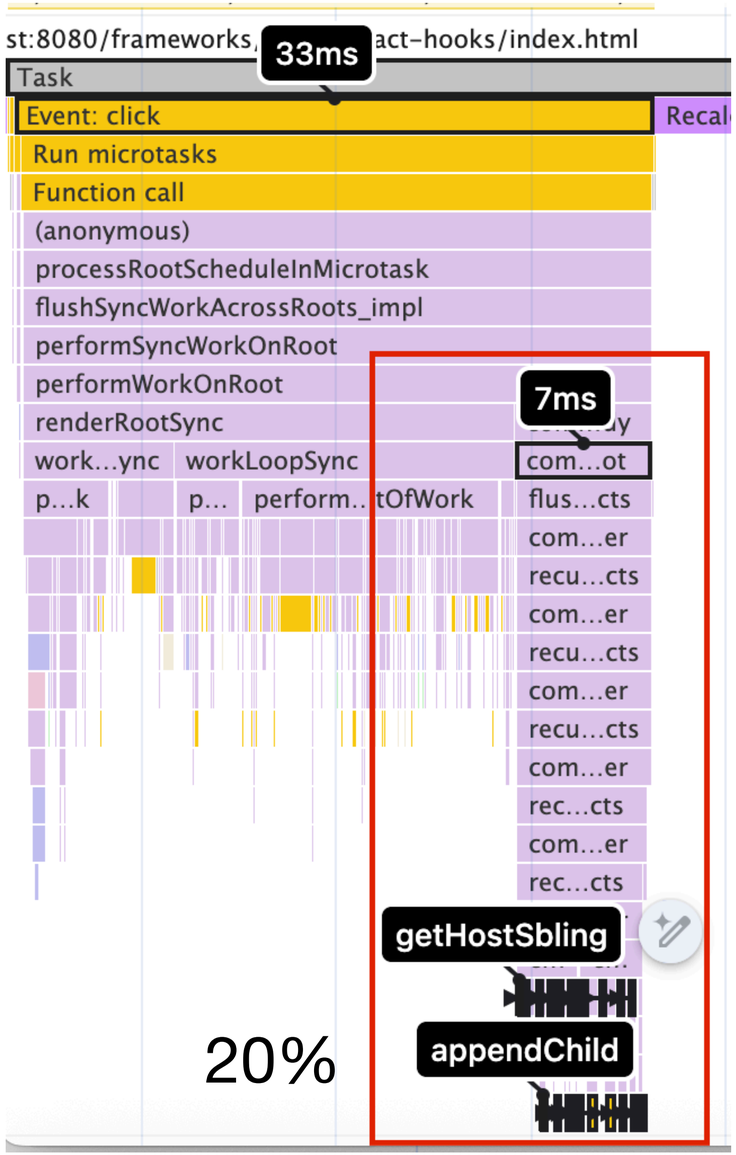


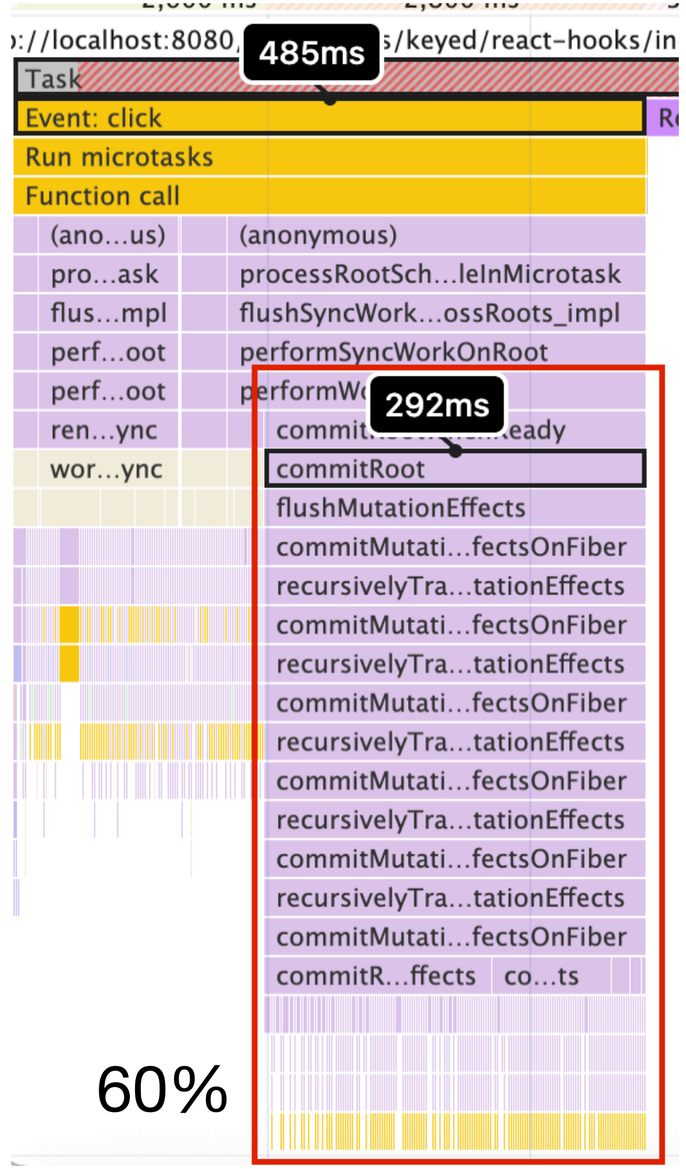
1000 строк



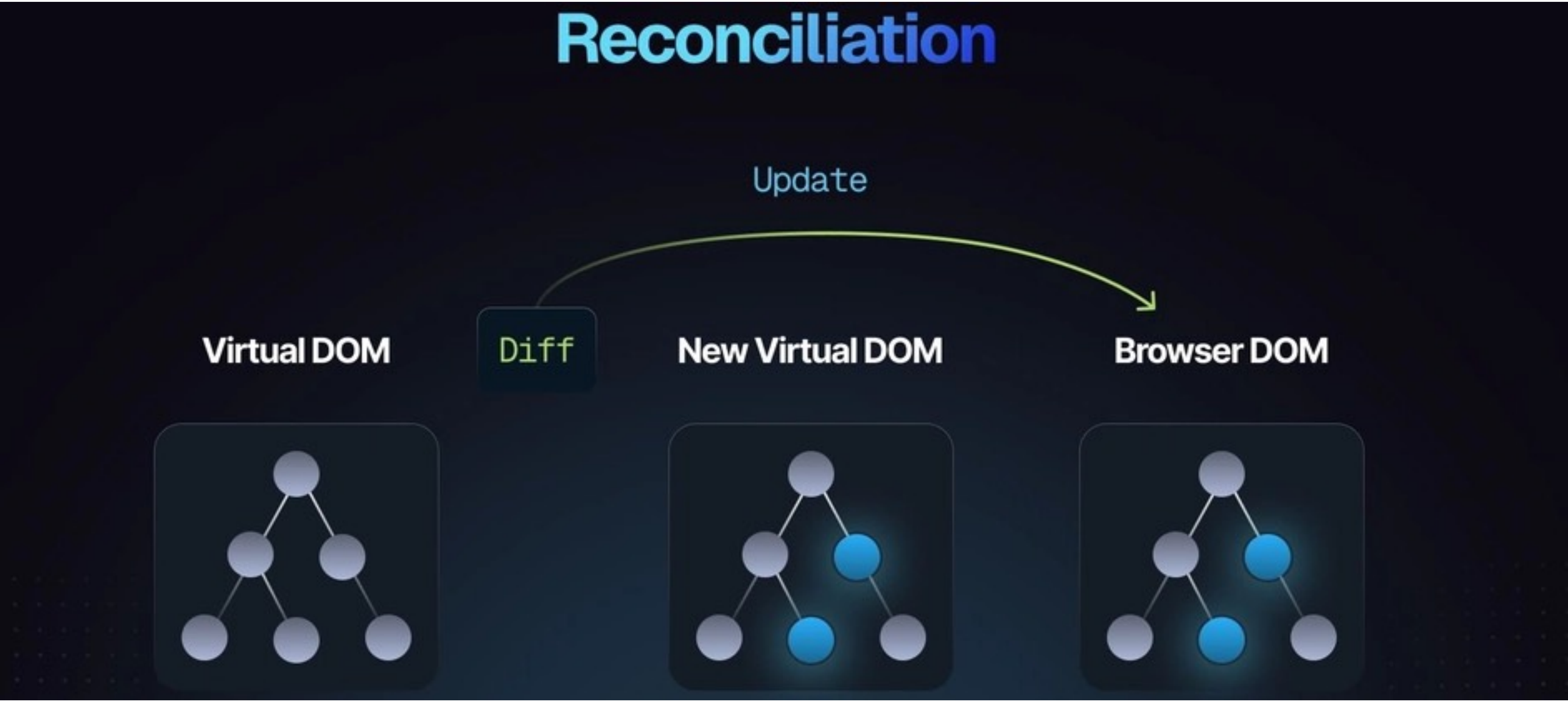
10 000 строк

Фаза коммит


st:8080/frameworks/
act-hooks/index.html
33ms
Task
Event: click
Recal
Run microtasks
Function call
(anonymous)
processRootScheduleInMicrotask
flushSyncWorkAcrossRoots_impl
performSyncWorkOnRoot
performWorkOnRoot
renderRootSync
work…ync
workLoopSync
p…k
p…
perform…
tOfWork
7ms
com…ot
flus…cts
com…er
recu…cts
com…er
recu…cts
com…er
recu…cts
com…er
rec…cts
com…er
rec…cts
getHostSbling
appendChild
20%

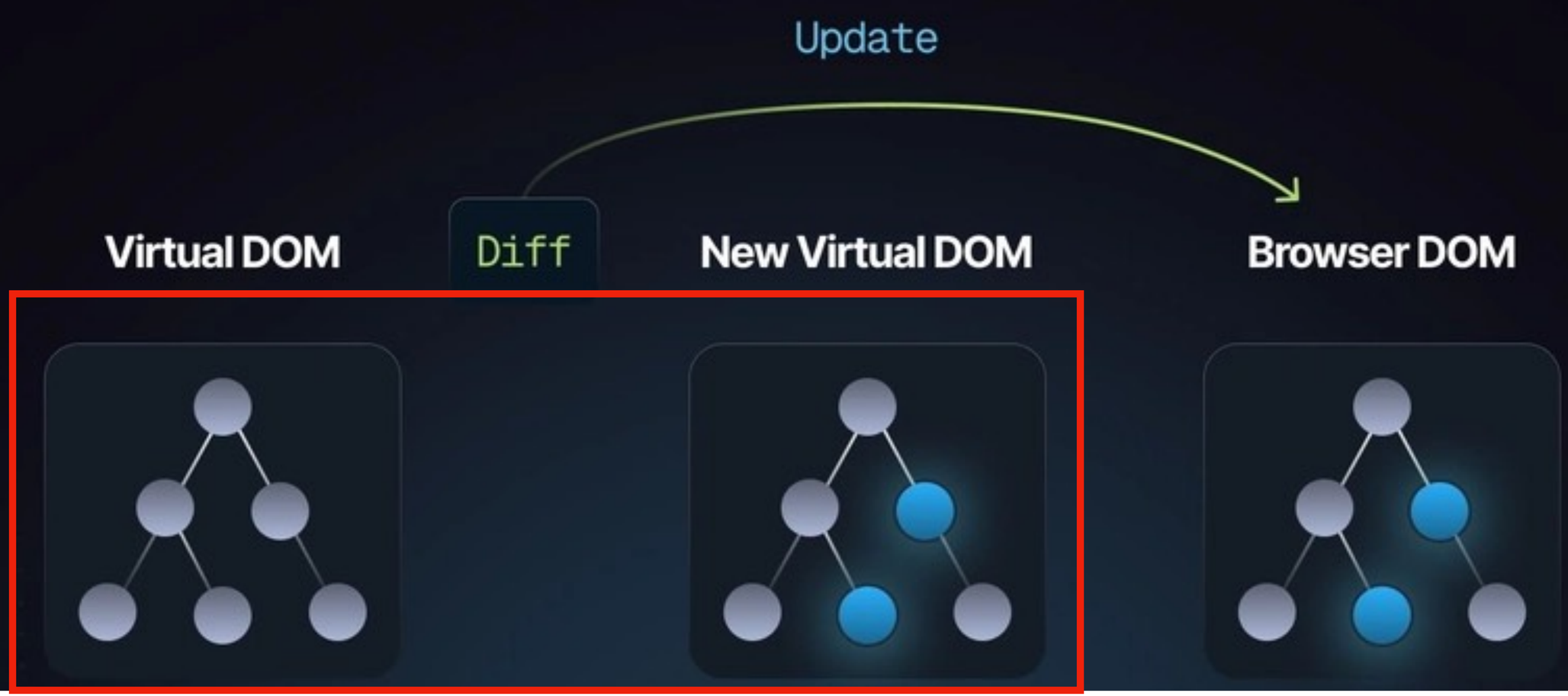

://localhost:8080/
s/keyed/react-hooks/in
485ms
Task
Event: click
Run microtasks
Function call
(ano…us)
(anonymous)
pro…ask
processRootSch…leInMicrotask
flus…mpl
flushSyncWork…ossRoots_impl
perf…oot
performSyncWorkOnRoot
perf…oot
ren…ync
wor…ync
292ms
commitRoot
flushMutationEffects
commitMutati…fectsOnFiber
recursivelyTra…tationEffects
commitMutati…fectsOnFiber
recursivelyTra…tationEffects
commitMutati…fectsOnFiber
recursivelyTra…tationEffects
commitMutati…fectsOnFiber
recursivelyTra…tationEffects
commitMutati…fectsOnFiber
recursivelyTra…tationEffects
commitMutati…fectsOnFiber
commitR…ffects
co…ts
60%

Что произошло в фазе Commit ?

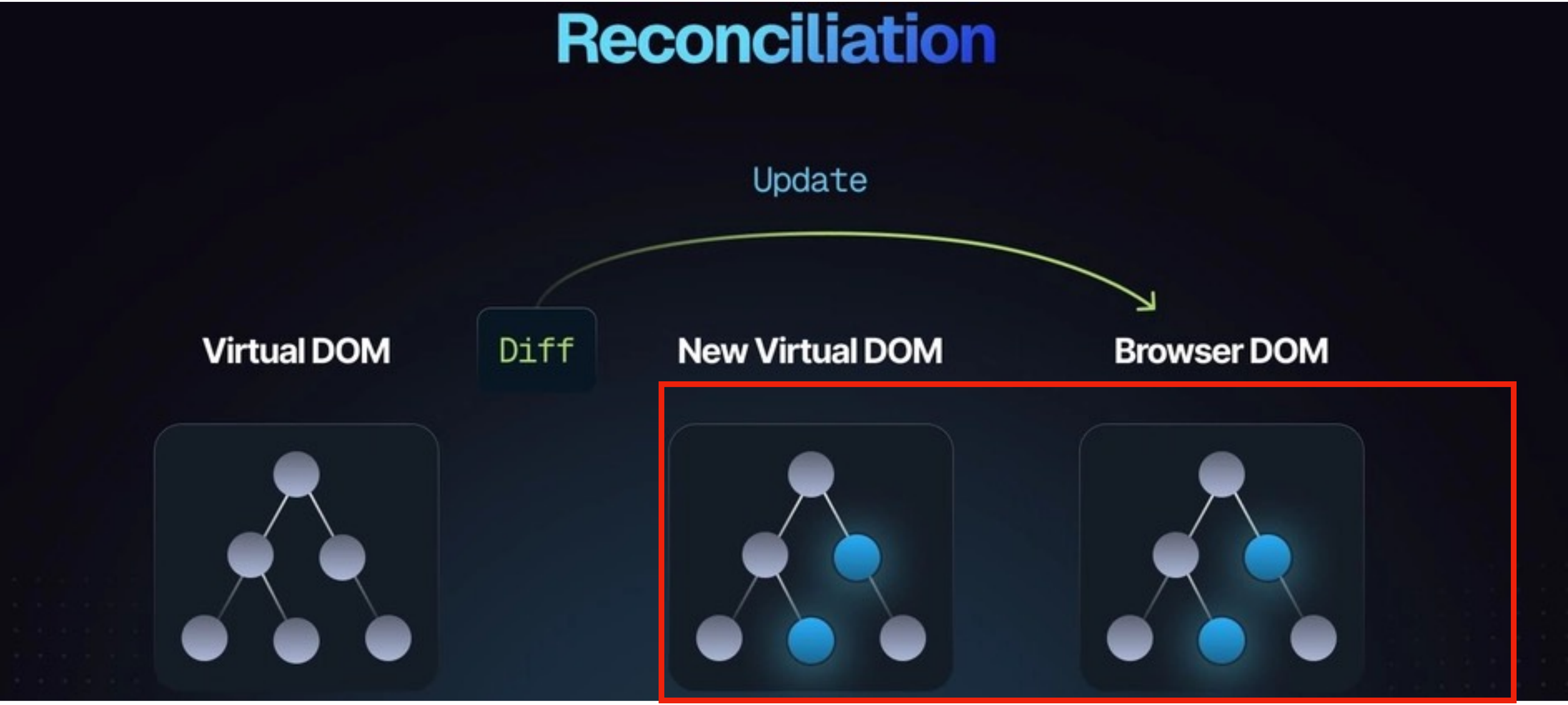

Reconciliation
Update
Virtual DOM
Diff
New Virtual DOM
Browser DOM

dev.to

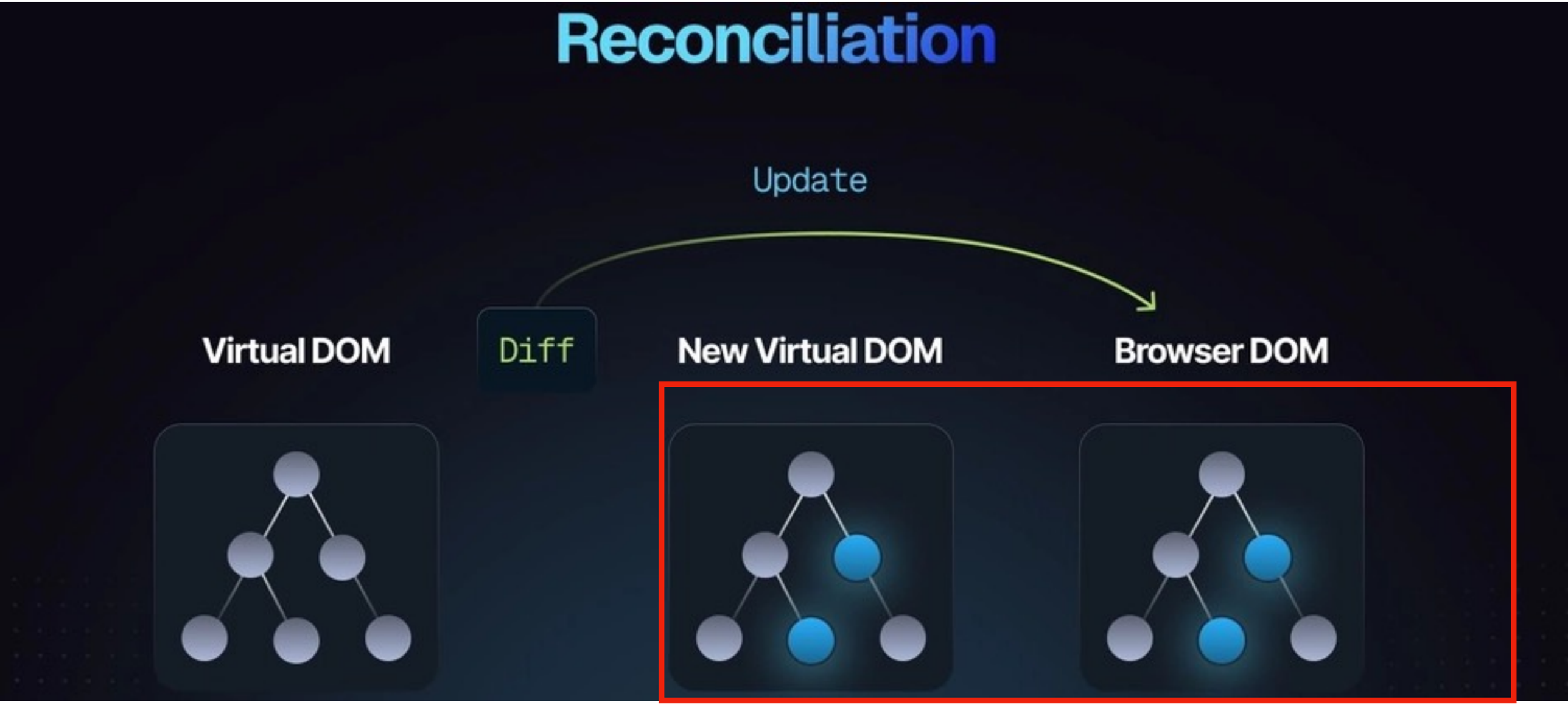
Reconciliation



dev.to


Reconciliation
Update
Virtual DOM
Diff
New Virtual DOM
Browser DOM

dev.to


Reconciliation
Update
Virtual DOM
Diff
New Virtual DOM
Browser DOM

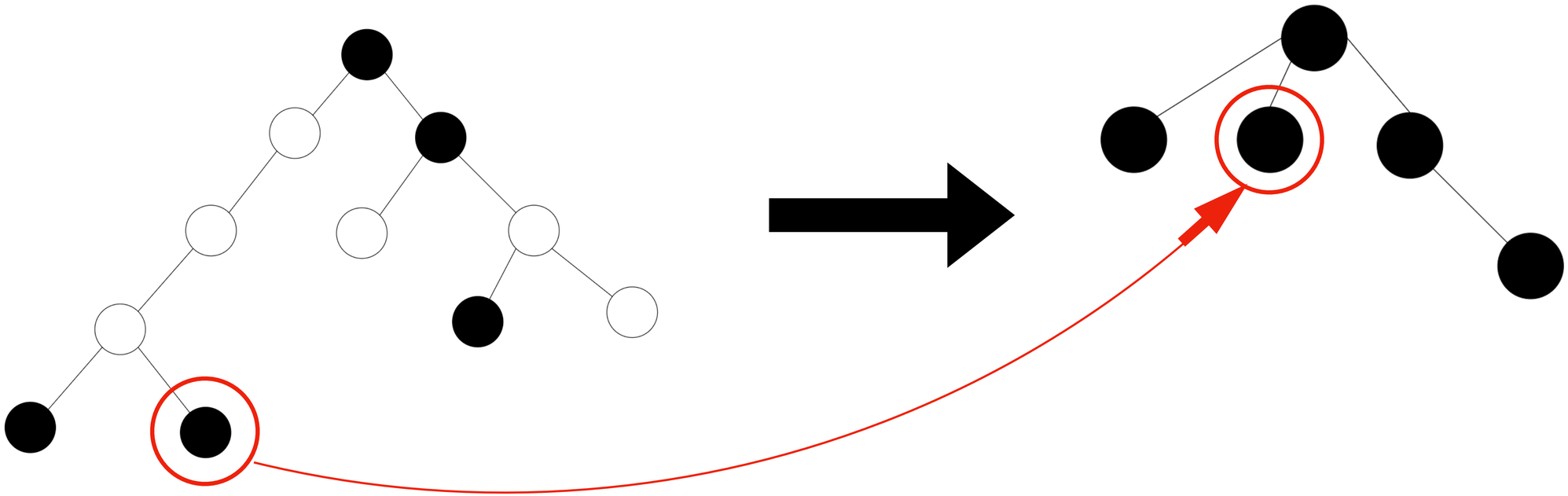
dev.to

Host компоненты (div, span, p, …)

Функциональные компоненты

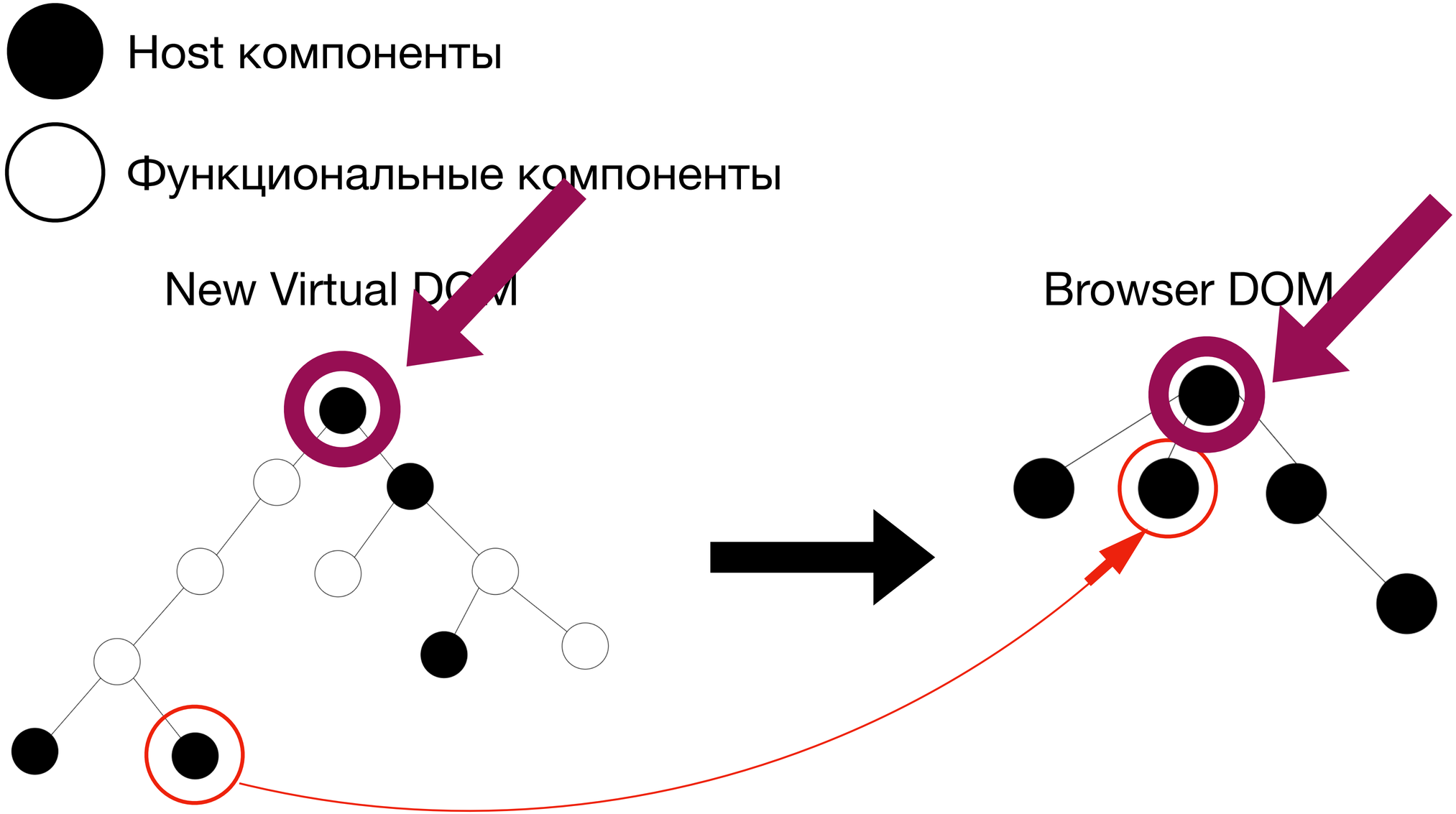
New Virtual DOM

Browser DOM



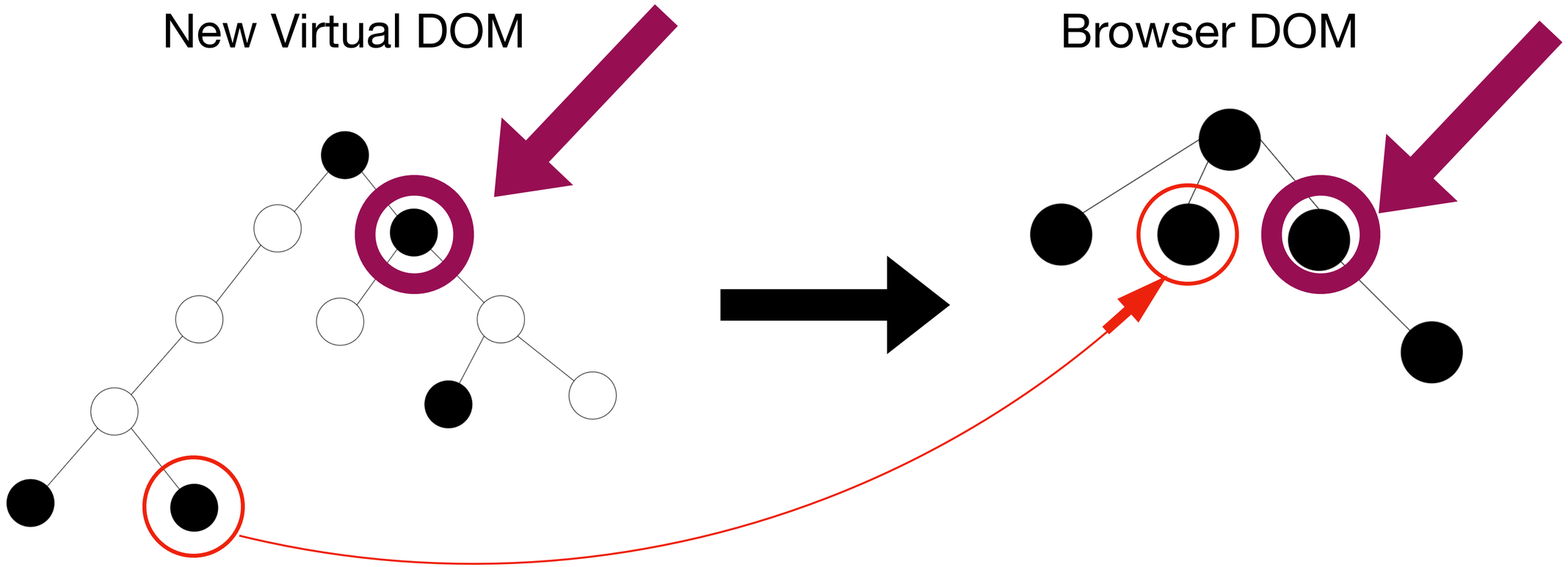
Host компонент (browser DOM)

- Найти в vdom родителя (parent) типа host
- Найти в vdom ближайший соседний (sibling) элемент типа хост
- parent.insertBefore(«наш host компонент», sibling) или parent.appendChild(«наш host компонент») если sibling отсутствует


Host компоненты
Функциональные компоненты
New Virtual DOM
Browser DOM

Host компоненты

Функциональные компоненты



Host компонент (browser DOM)

- Найти в vdom родителя (parent) типа host
- Найти в vdom ближайший соседний (sibling) элемент типа хост

getHostSibling

- parent.insertBefore(«наш host компонент», sibling) или parent.appendChild(«наш host компонент») если sibling отсутствует

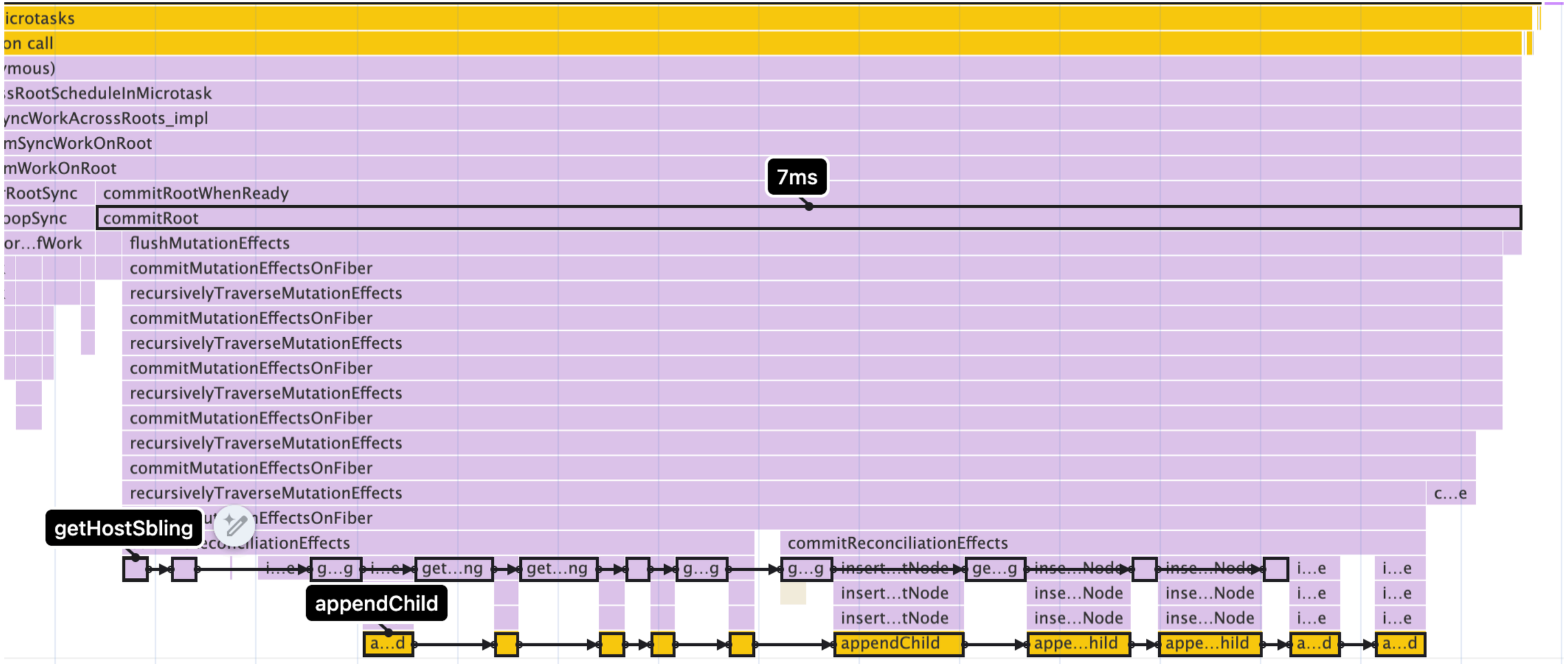
Host компонент (browser DOM)

- Найти в vdom родителя (parent) типа host
- Найти в vdom ближайший соседний (sibling) элемент типа хост

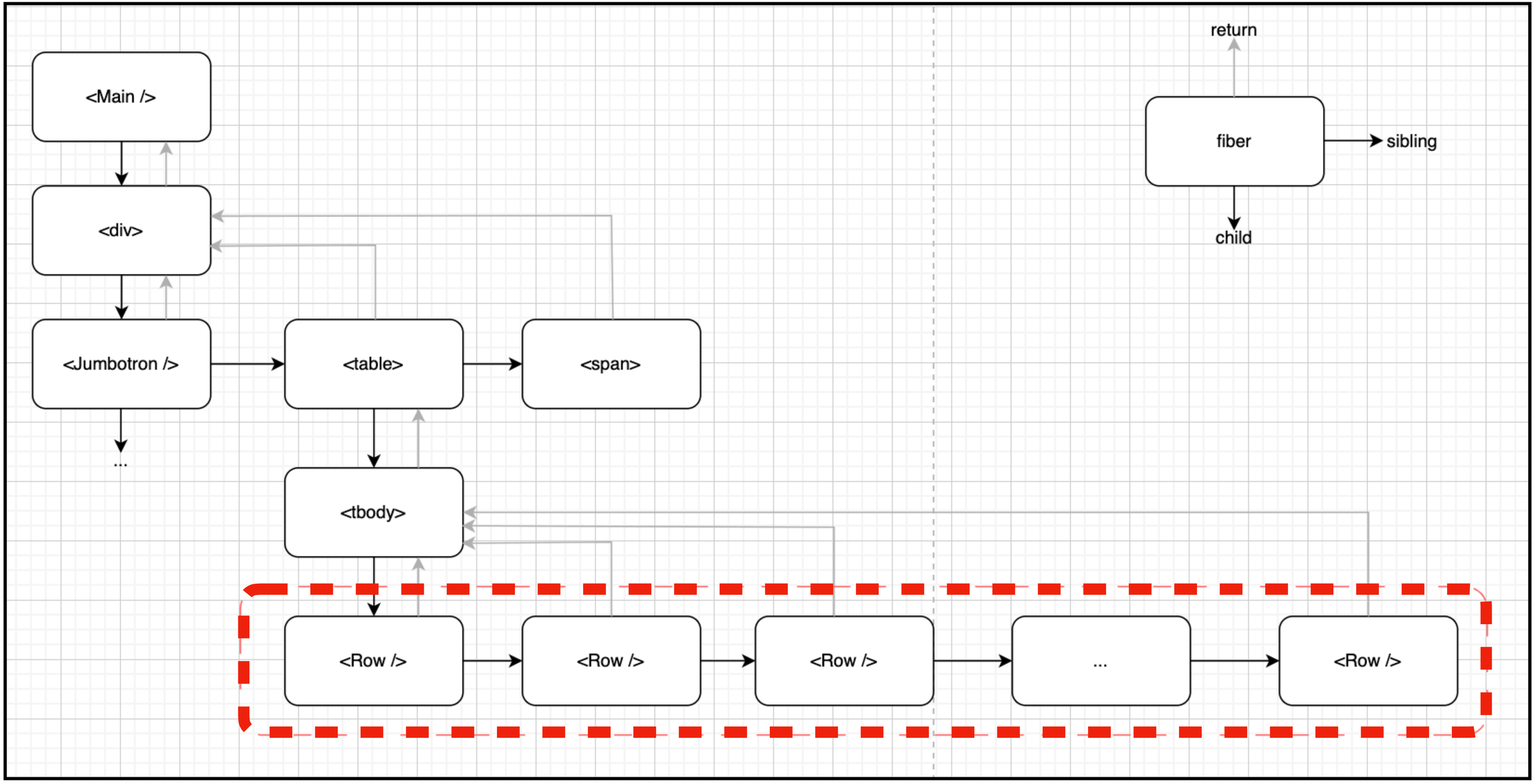
getHostSibling

- parent.insertBefore(«наш host компонент», sibling) или parent.appendChild(«наш host компонент») если sibling отсутствует

Create 1000 rows

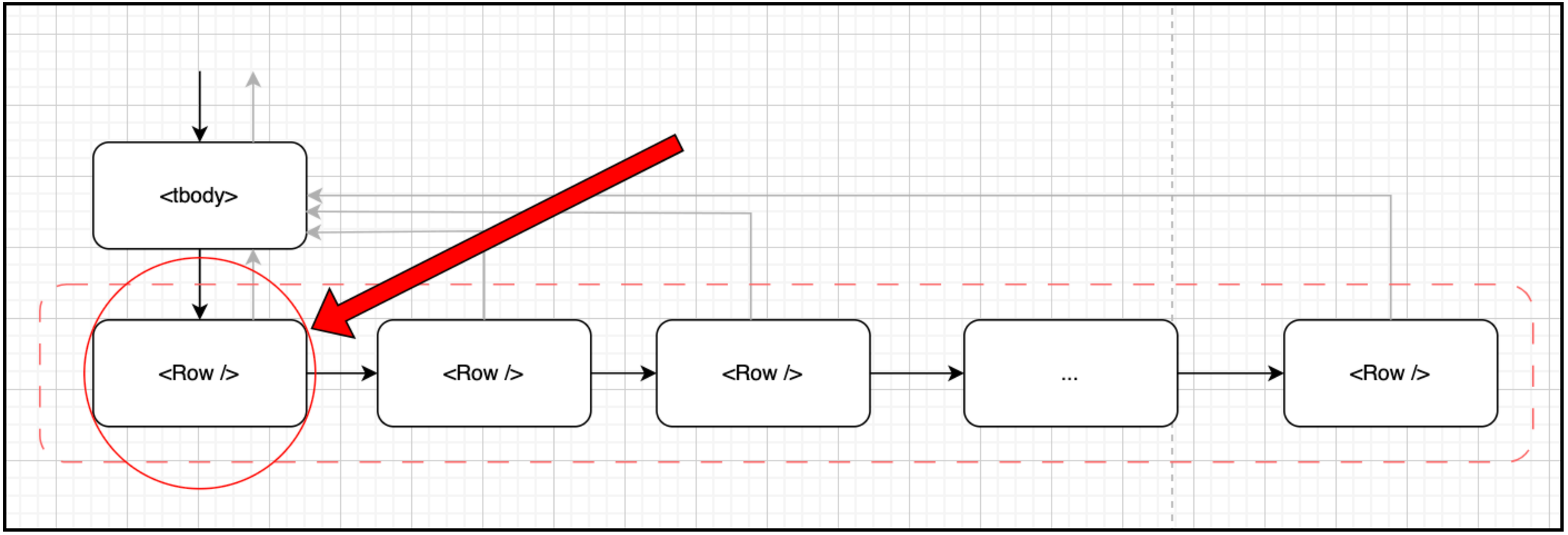


```
const Main = () => {
  const [{ data, selected }, dispatch] = useReducer(listReducer, initialState);

  return (<div className="container">
    <Jumbotron dispatch={dispatch} />
    <table className="table table-hover table-striped test-data">
      <tbody>
        {data.map(item => (
          <Row key={item.id} item={item} selected={selected === item.id} dispatch={dispatch} />
        ))}
      </tbody>
    </table>
    <span className="preloadicon glyphicon glyphicon-remove" aria-hidden="true" />
  </div>);
}
```


return
fiber
sibling
child
<Main />

<Jumbotron />
...
<table>

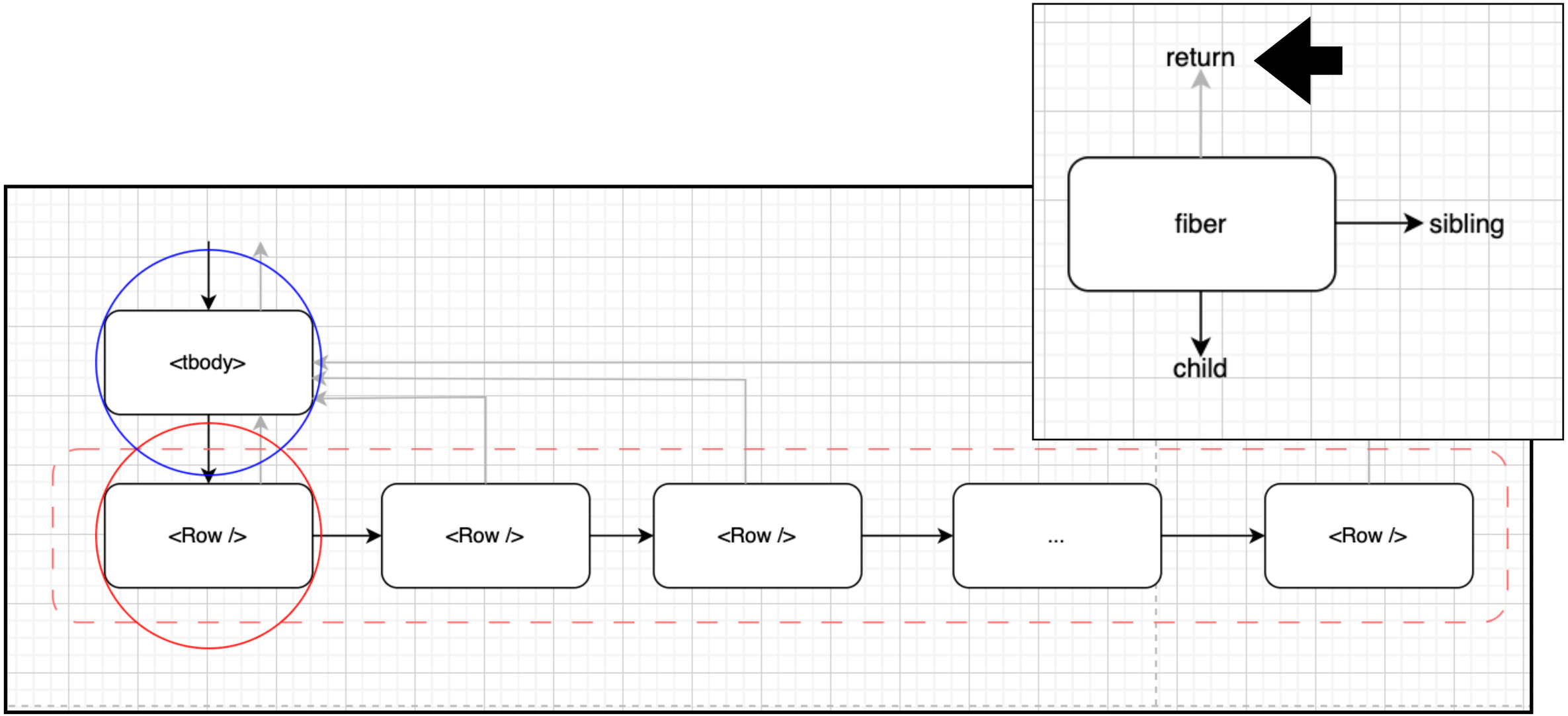
<tbody>
<Row />
<Row />
<Row />
...
<Row />


<tbody>
<Row />
<Row />
<Row />
...
<Row />

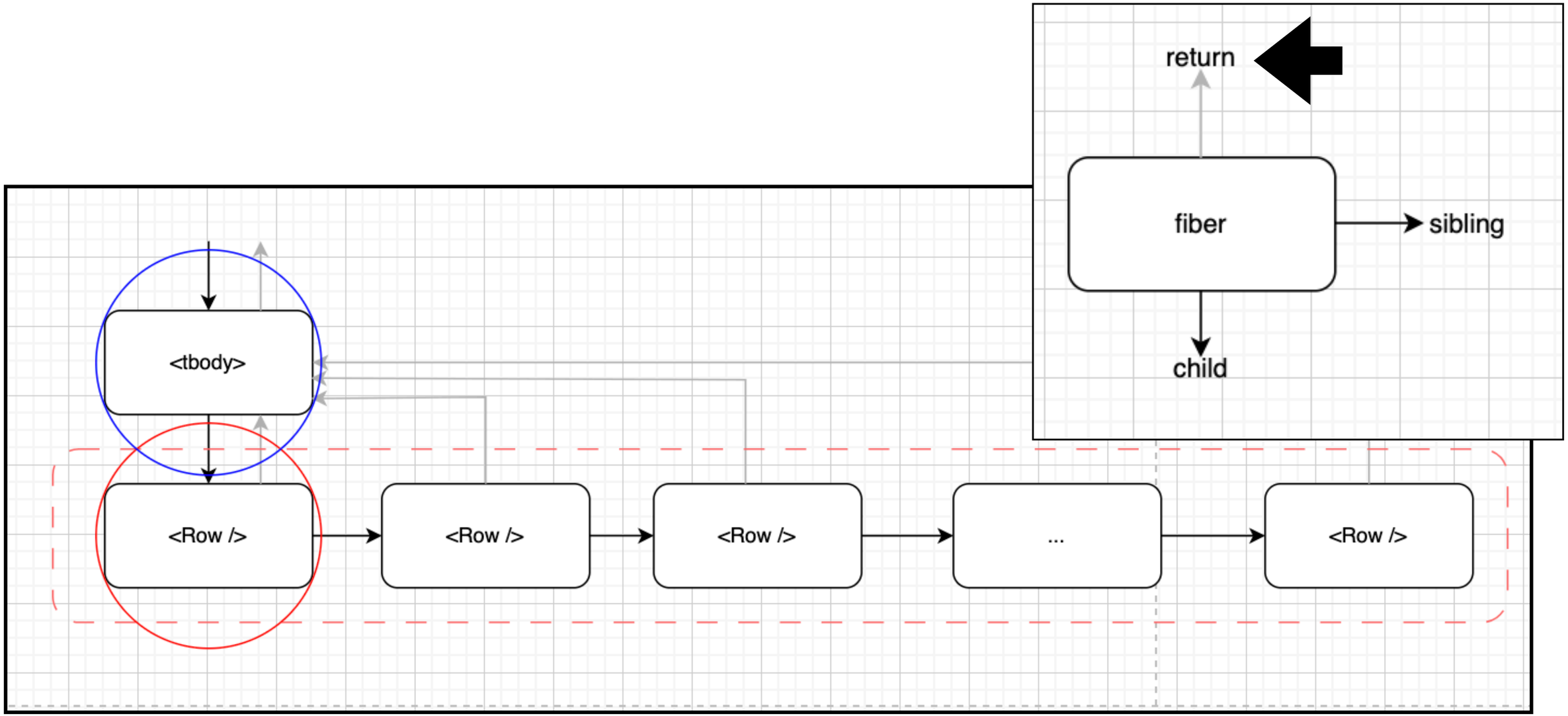
Host компонент (browser DOM)

- Найти в vdom родителя (parent) типа host
- Найти в vdom ближайший соседний (sibling) элемент типа хост
- parent.insertBefore(«наш host компонент», sibling) или parent.appendChild(«наш host компонент») если sibling отсутствует

Шаг 1 - Найти в vdom родителя (parent) типа host



Шаг 1 - Найти в vdom родителя (parent) типа host



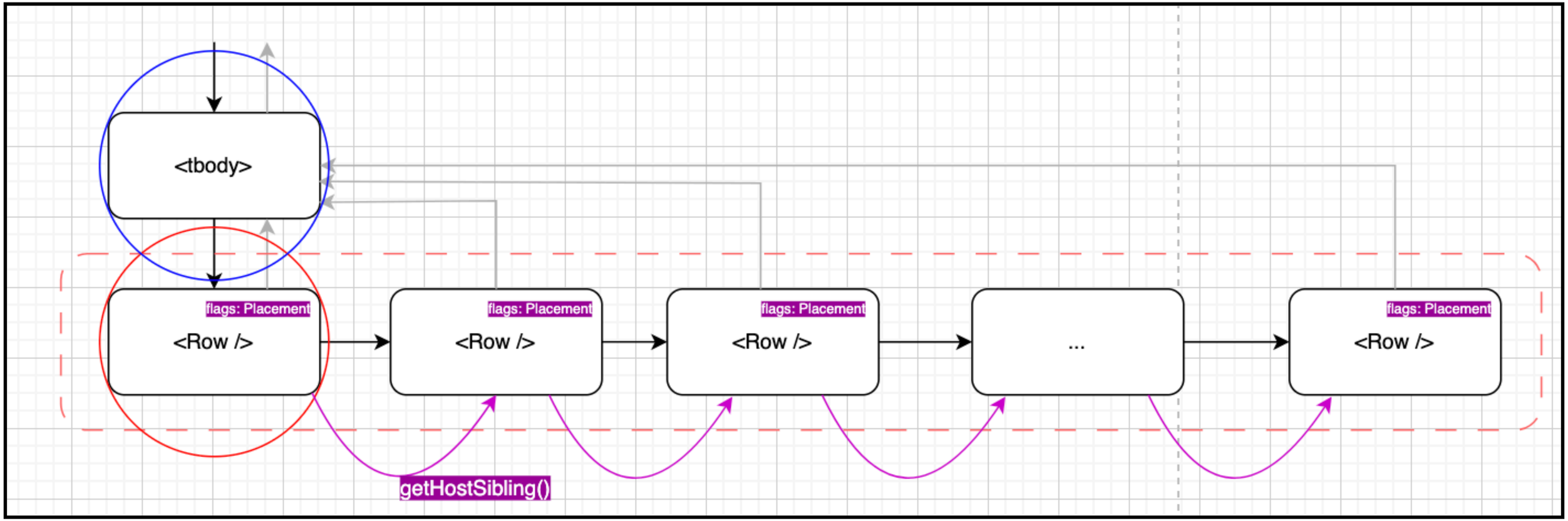
Host компонент (browser DOM)

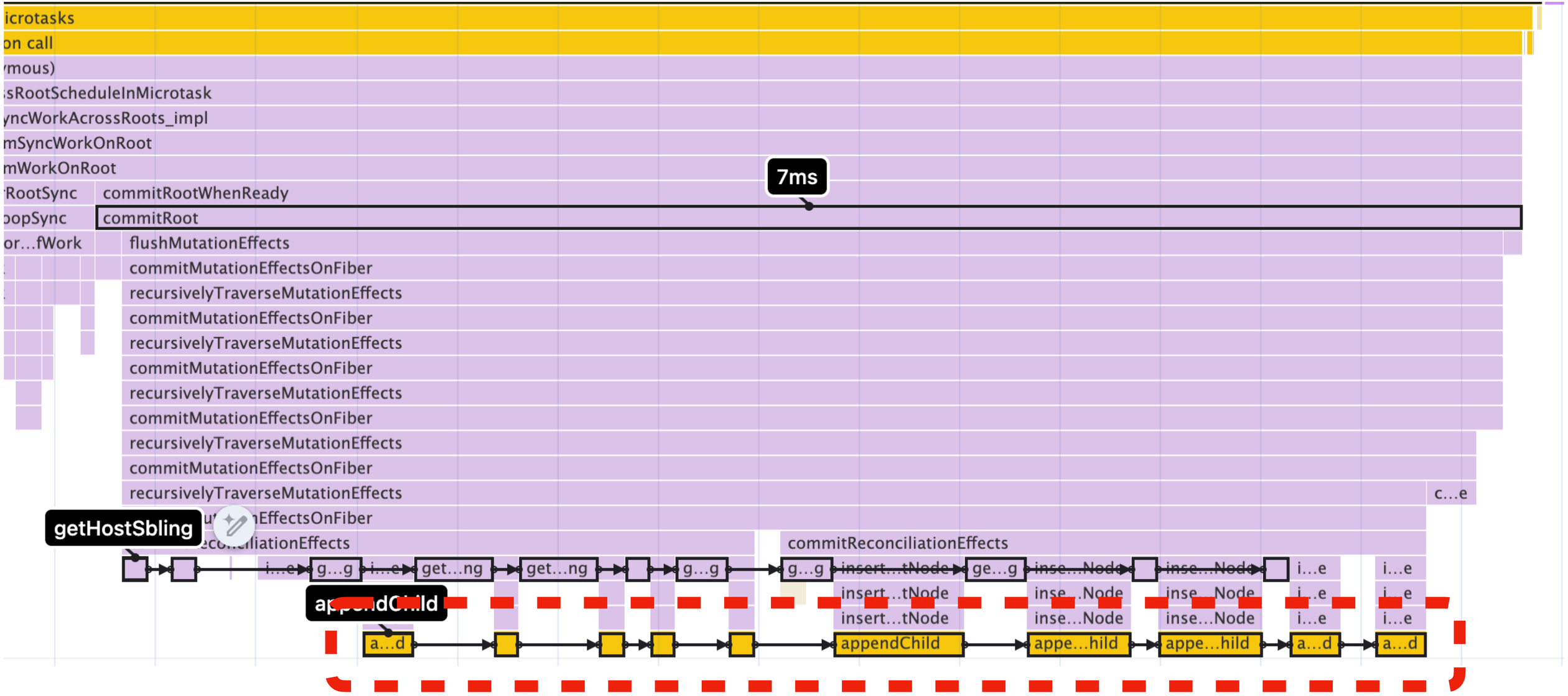
- Найти в vdom родителя (parent) типа host
- Найти в vdom ближайший соседний (sibling) элемент типа хост

getHostSibling

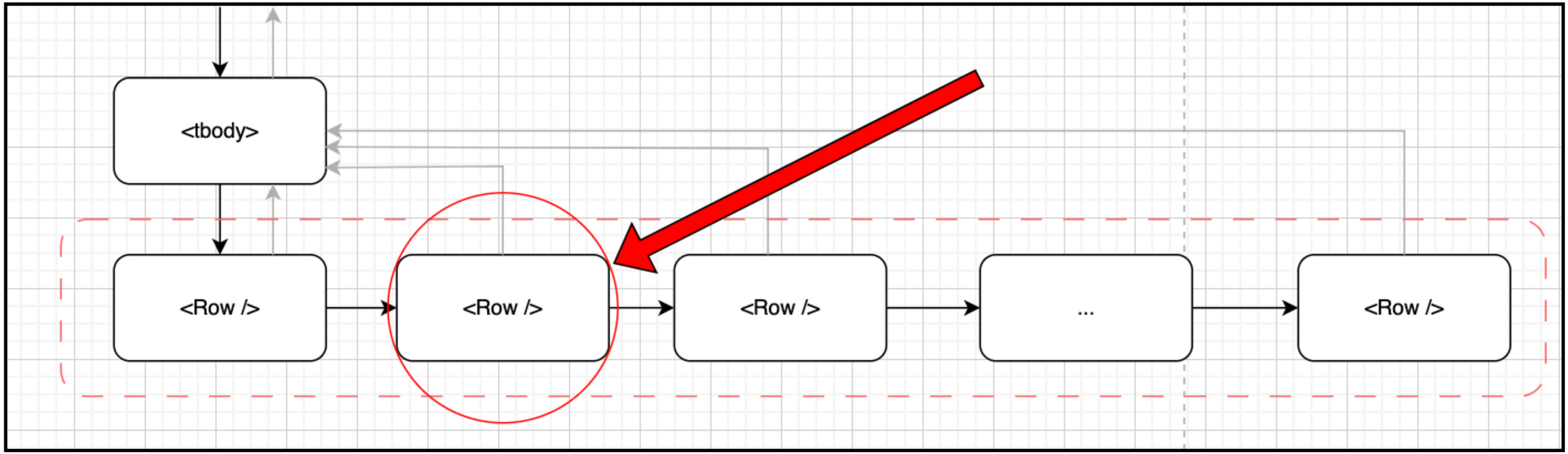
- parent.insertBefore(«наш host компонент», sibling) или parent.appendChild(«наш host компонент») если sibling отсутствует

Найти в vdom ближайший соседний (sibling) элемент типа хост

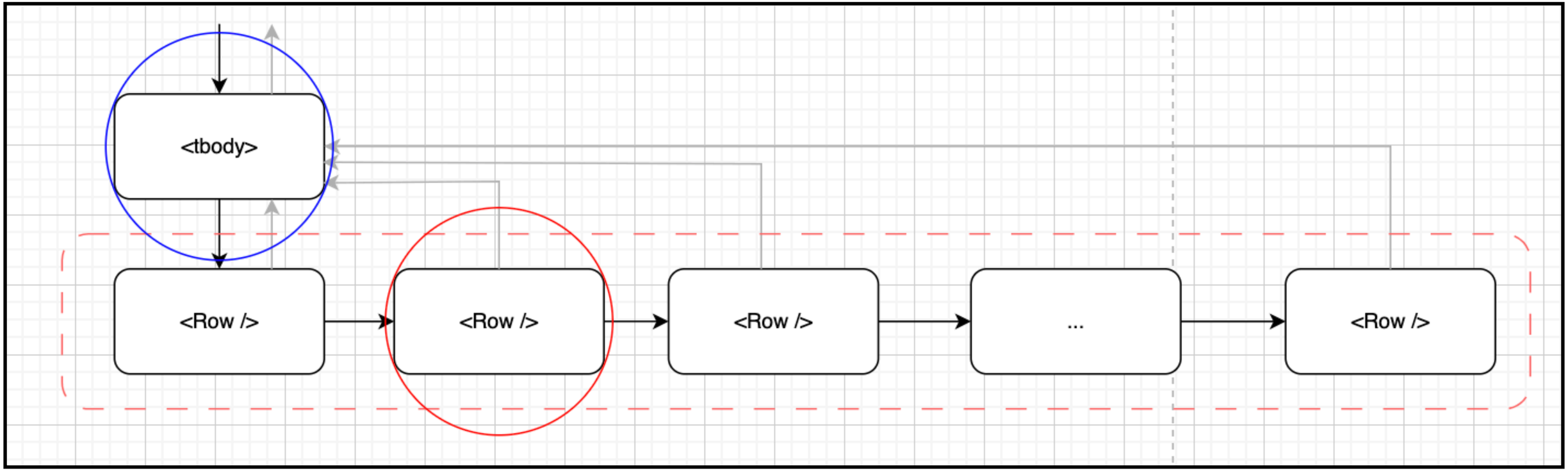



icrotasks
on call
mous)
sRootScheduleInMicrotask
yncWorkAcrossRoots_impl
mSyncWorkOnRoot
mWorkOnRoot
RootSync
commitRootWhenReady
7ms
oopSync
commitRoot
or...fWork
flushMutationEffects
commitMutationEffectsOnFiber
recursivelyTraverseMutationEffects
commitMutationEffectsOnFiber
recursivelyTraverseMutationEffects
commitMutationEffectsOnFiber
recursivelyTraverseMutationEffects
commitMutationEffectsOnFiber
recursivelyTraverseMutationEffects
commitMutationEffectsOnFiber
recursivelyTraverseMutationEffects
c...e
getHostSbling
commitReconciliationEffects
i...e
g...g
get...ng
insert...tNode
ge...g
inse...Node
appendChild
appe...hild
a...d

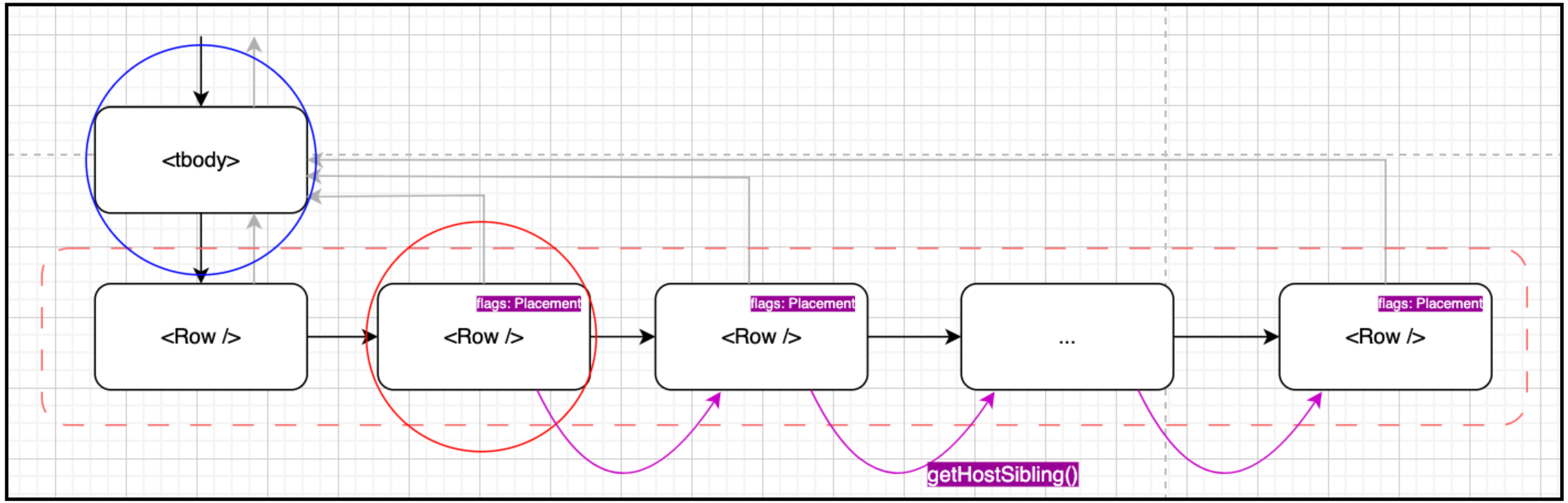
И так с каждым элементом массива



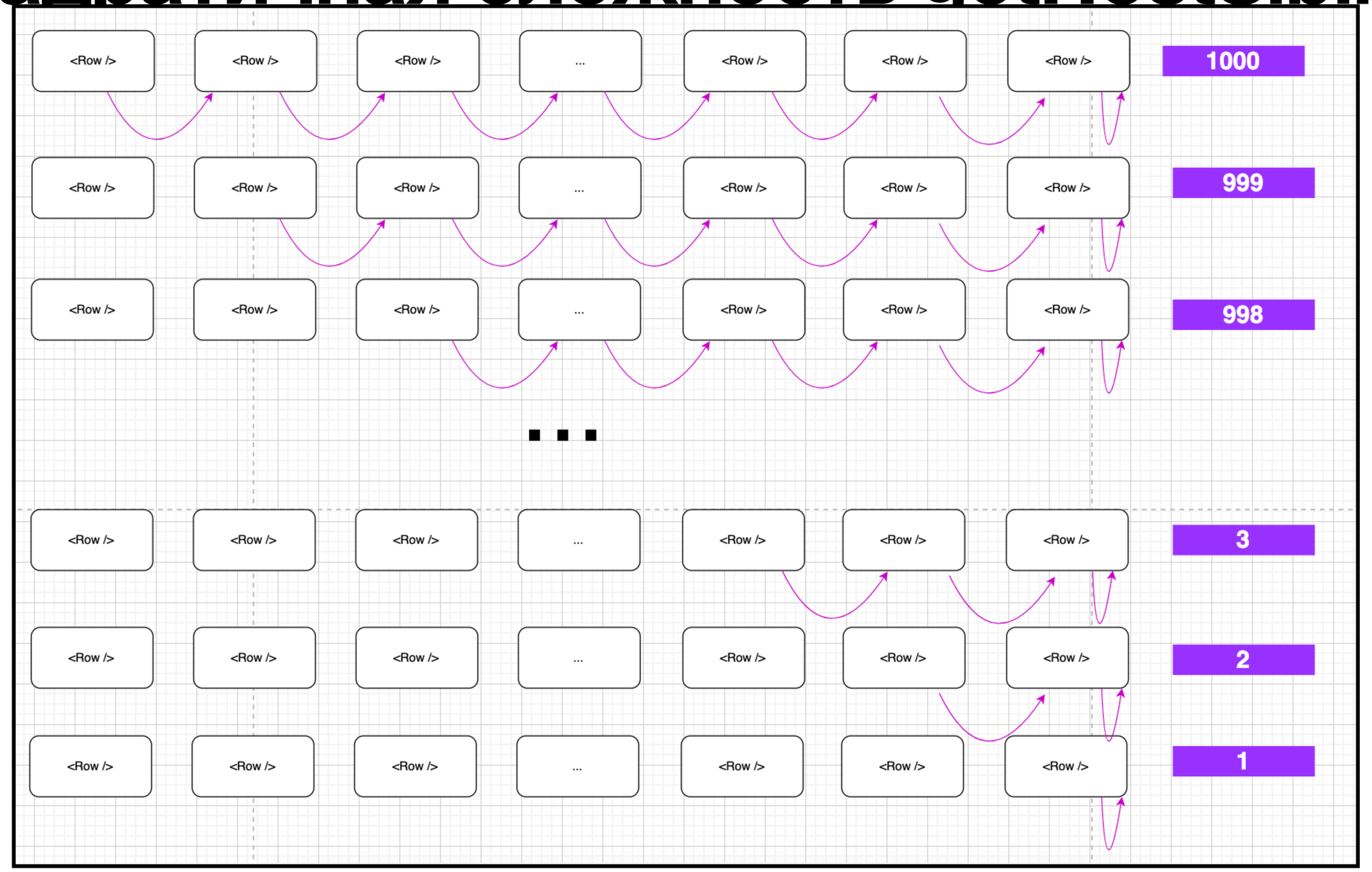
И так с каждым элементом массива



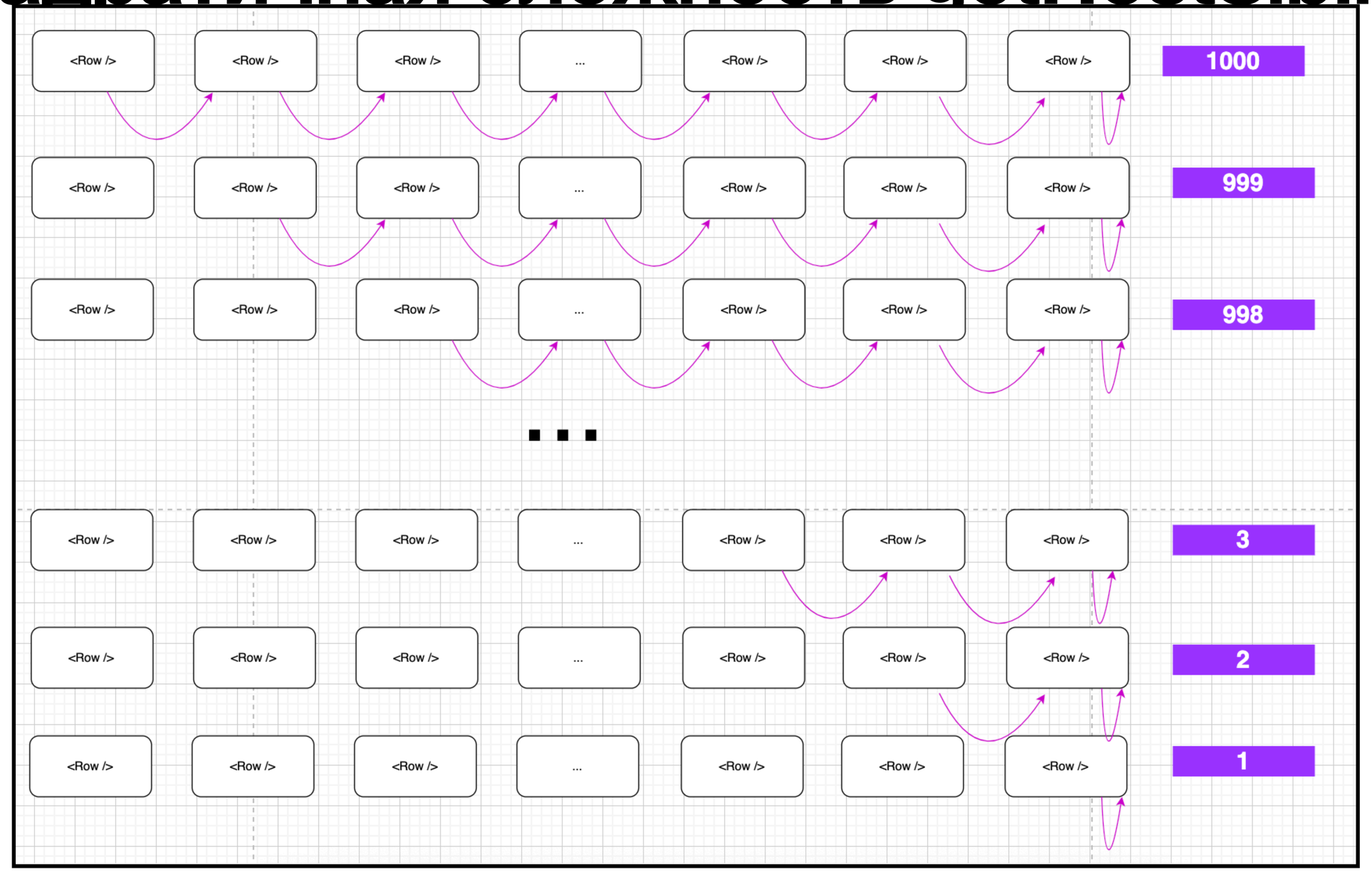
И так с каждым элементом массива



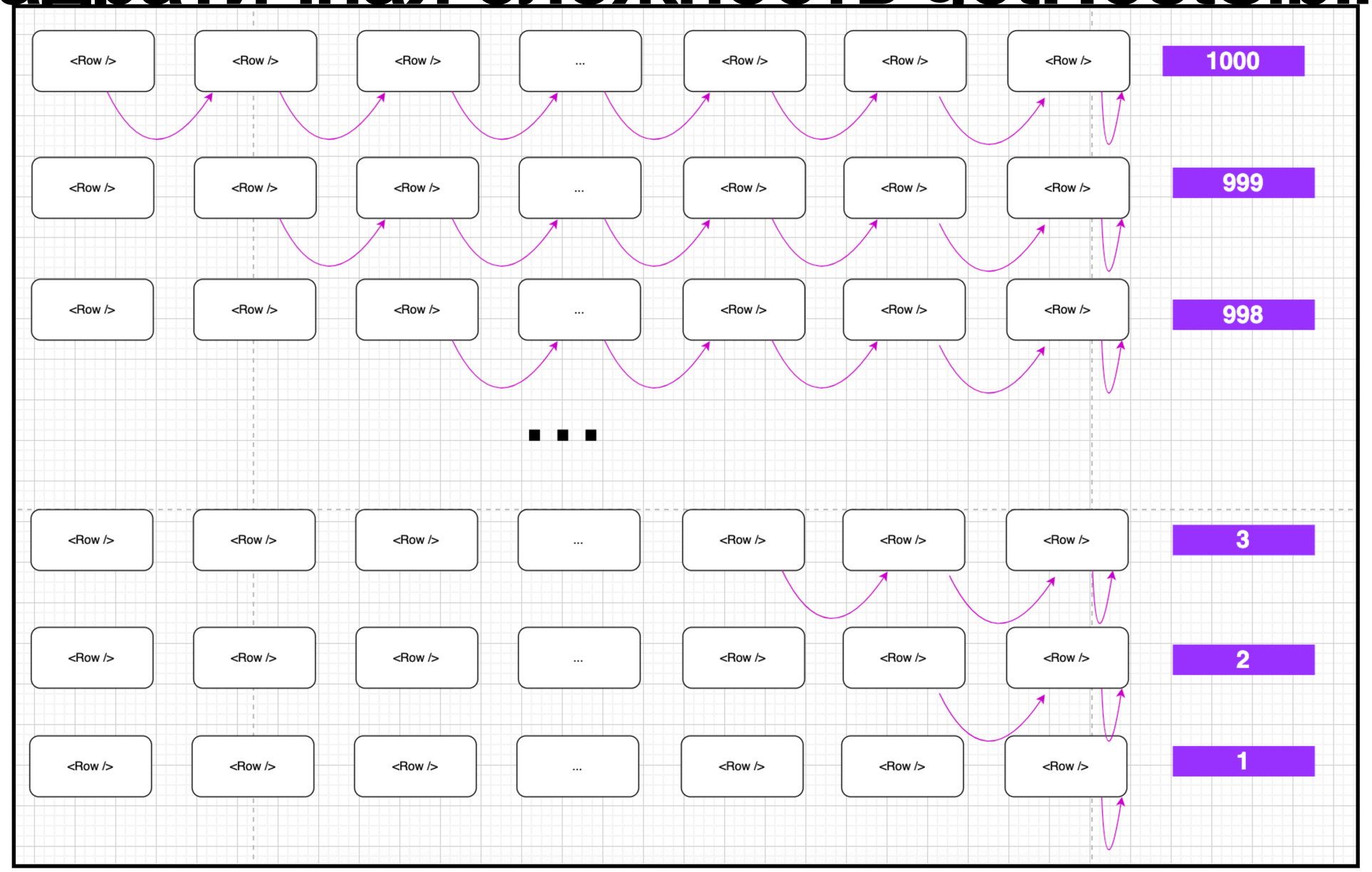
Квадратичная сложность getHostSibling



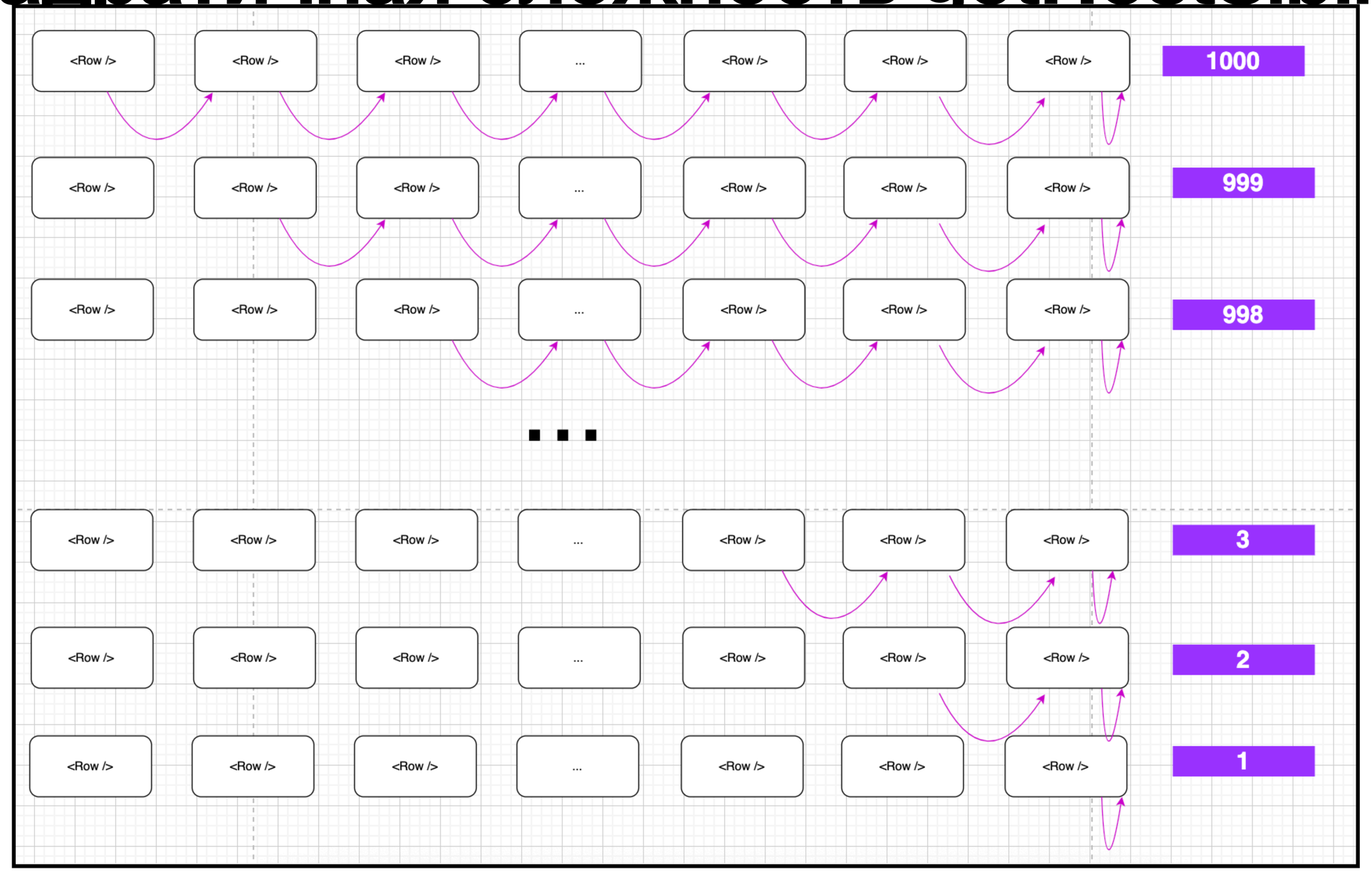
Квадратичная сложность getHostSibling



Квадратичная сложность getHostSibling



Квадратичная сложность getHostSibling



Квадратичная сложность getHostSibling

Арифметическая прогрессия $\frac{n(n+1)}{2}$

500 500 операций для n = 1000

50 005 000 операций для n = 10 000

X 100

Решение: объединить тысячи нод в одно поддерево

До

```
<tbody>
  {data.map(item => (
    <Row
    key={item.id}
    item={item}
    selected={selected === item.id}
    dispatch={dispatch} />
  ))}
</tbody>
```

На этапе render создаём тысячи отдельных нод Row

На этапе commit находим место в browser DOM для тысяч нод Row

После

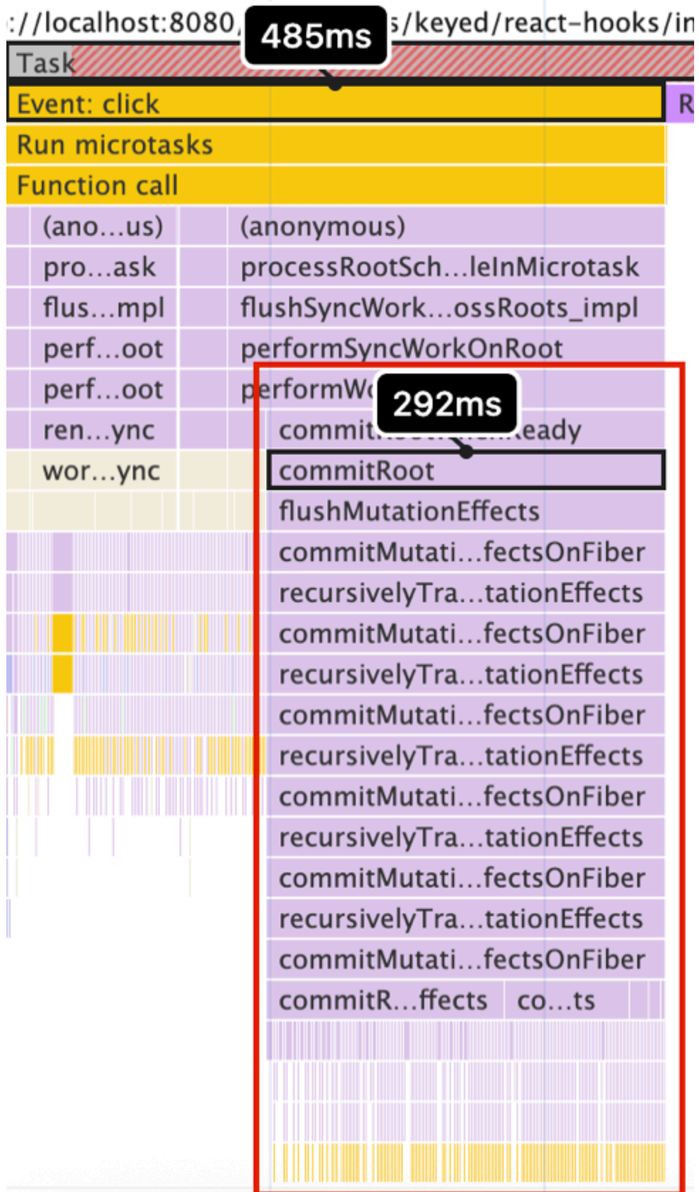
```
{!!data.length &&
    <tbody>
    {data.map(item => (
      <Row
        key={item.id}
        item={item}
        selected={selected === item.id}
        dispatch={dispatch} />
    ))}
    </tbody>
  }
```

На этапе render создаём поддерево с корнем <tbody>

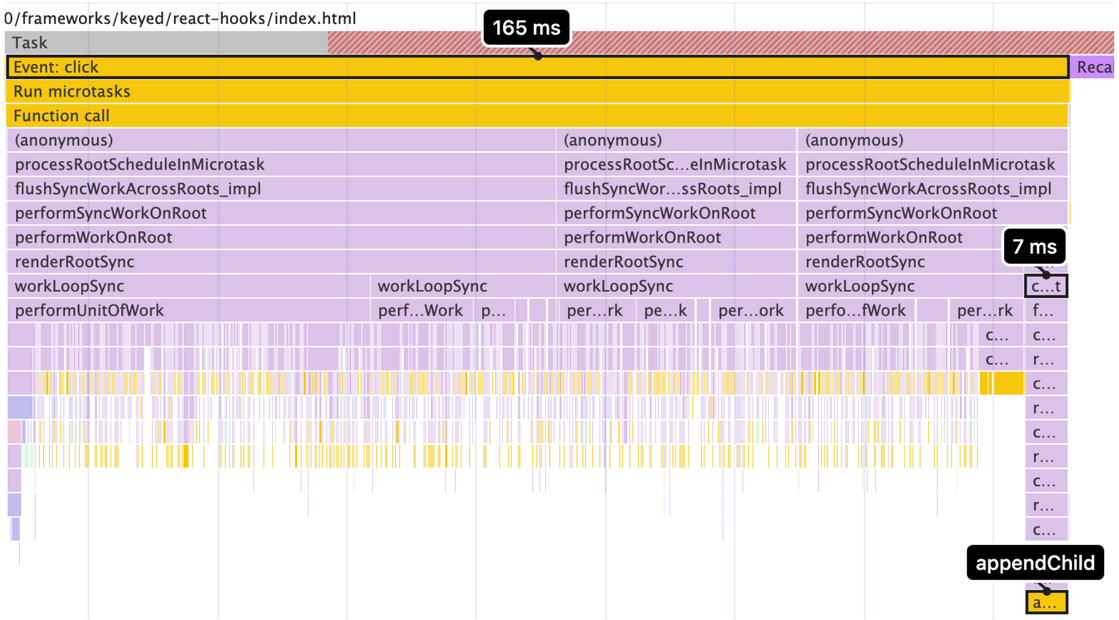
На этапе commit находим место в browser DOM только для <tbody>

Create 10 000 rows

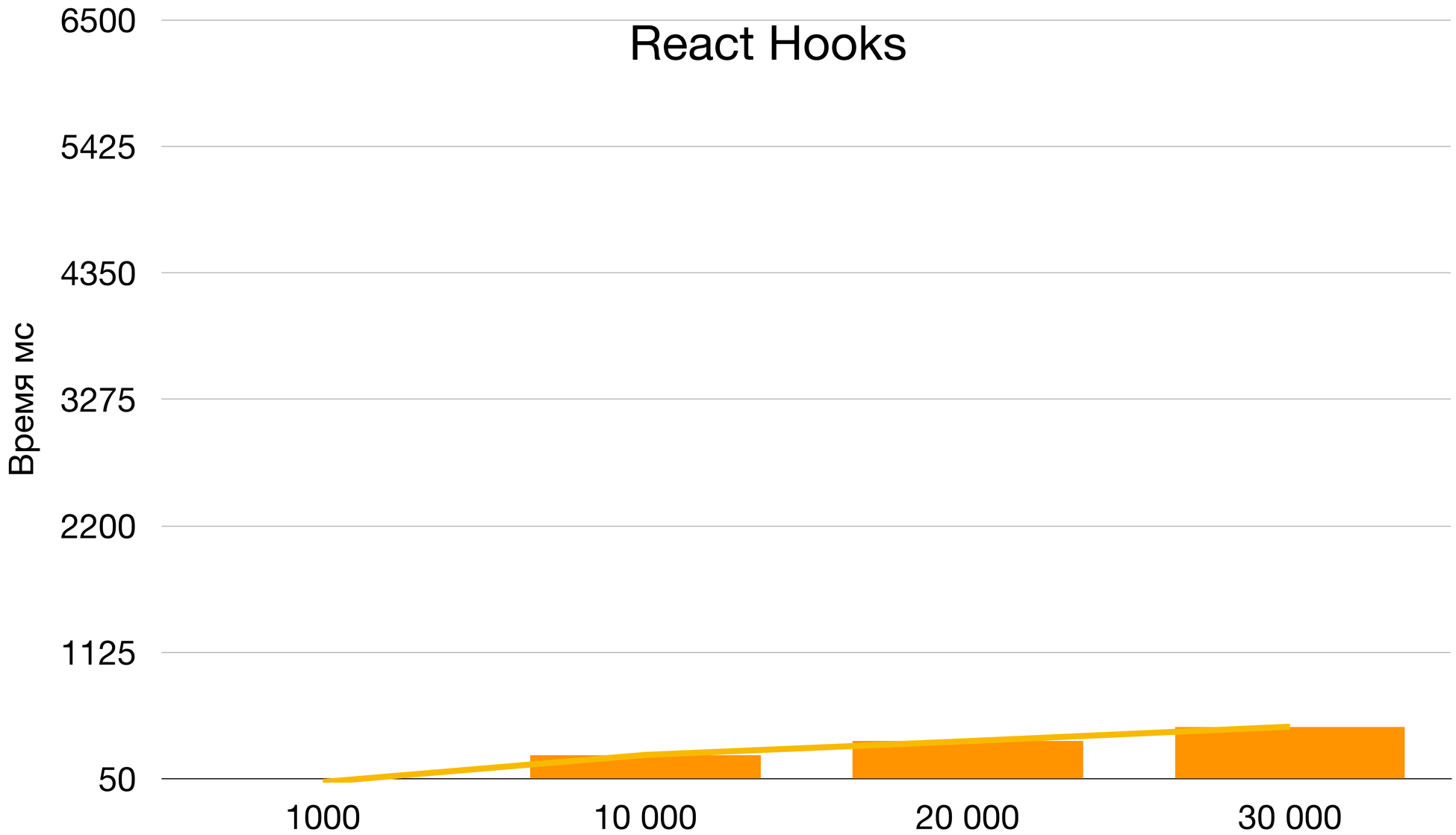
До



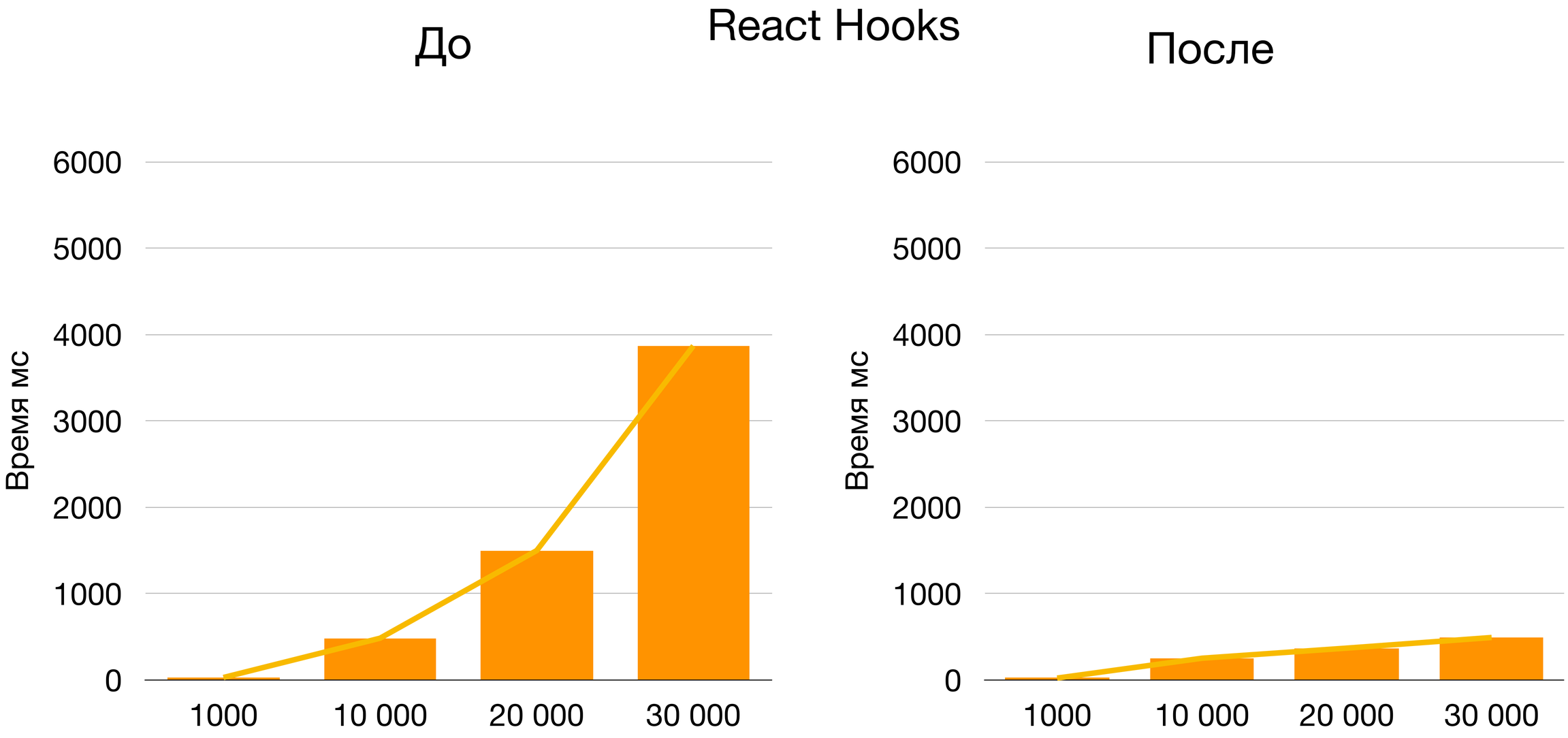
После



Create (many) rows

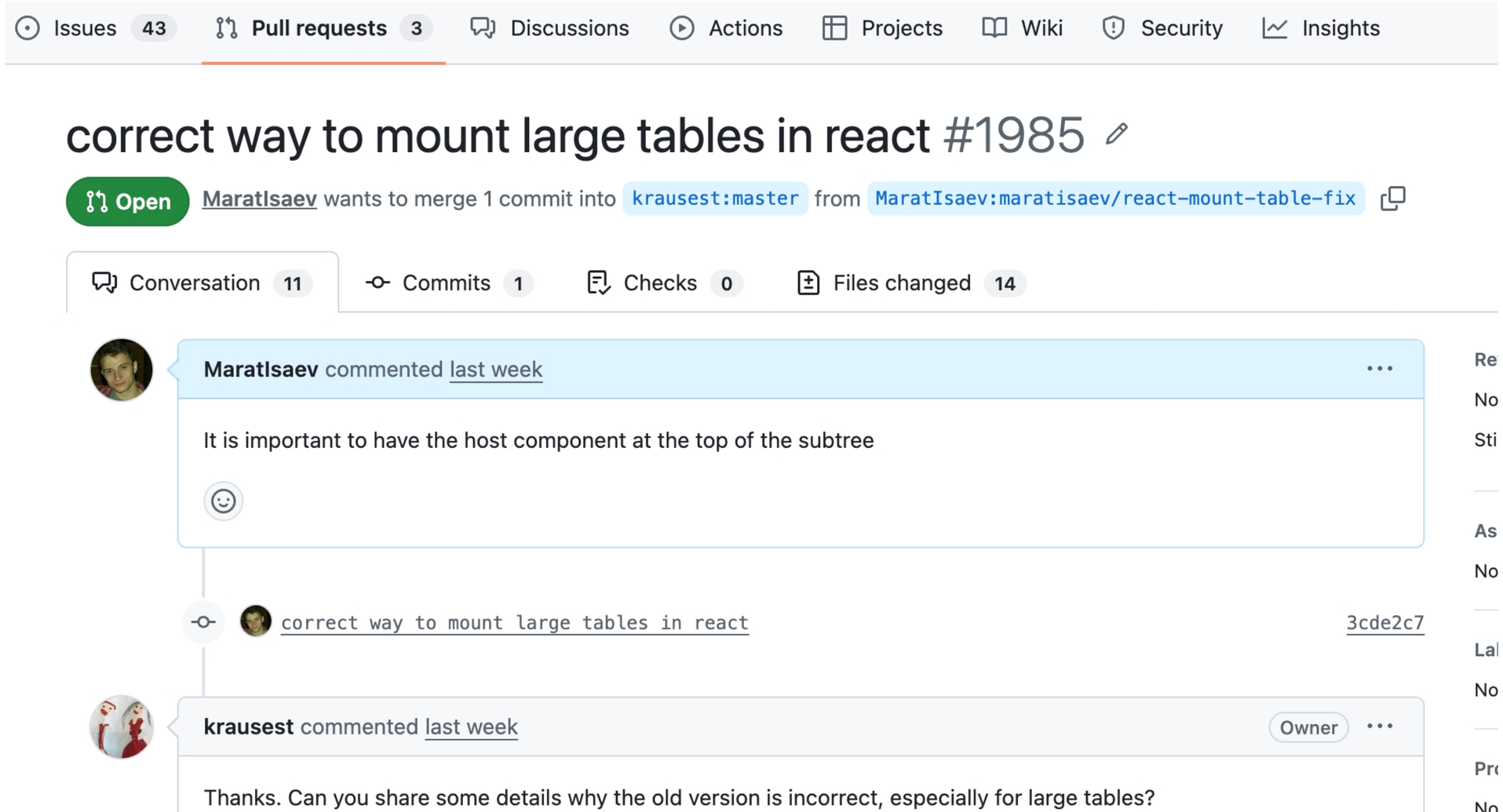


Create (many) rows



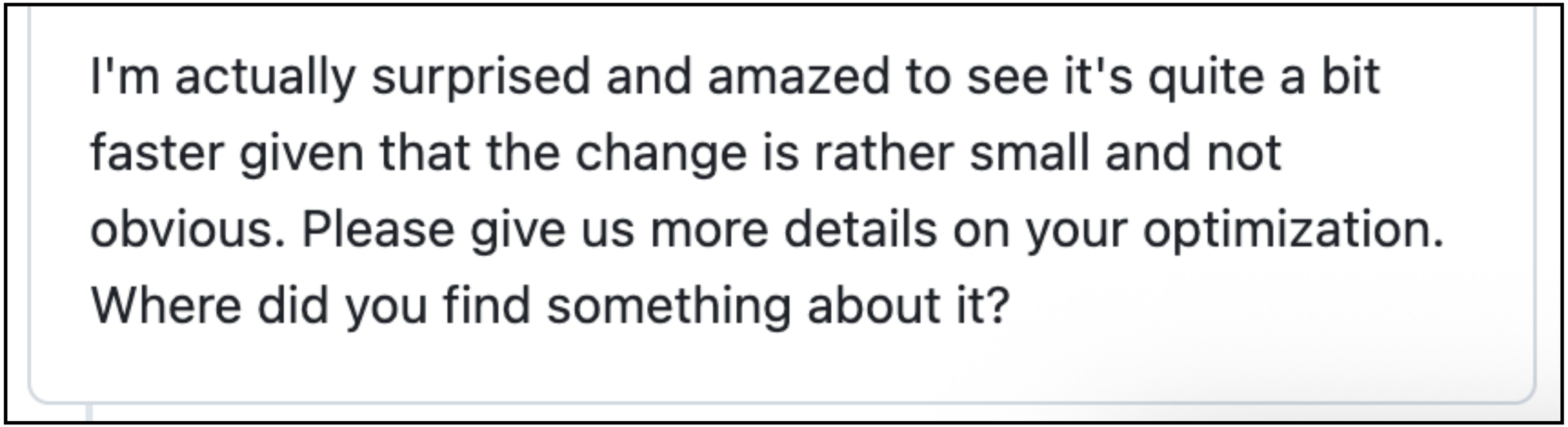
Create (many) rows: pull request

- https://github.com/krausest/js-framework-benchmark/pull/1985



Create (many) rows: pull request

- https://github.com/krausest/js-framework-benchmark/pull/1985



Create (many) rows: pull request

- https://github.com/krausest/js-framework-benchmark/pull/1985

"what is a user likely to write"

Create (many) rows

Solid

Svelte

?

Create (many) rows

SVELTE • TEMPLATE SYNTAX

{#each ...}

SEE ALSO

Tutorial • Basic Svelte • Logic • Each blocks

<For each={...} />

```
{#each data as row (row)}
    <tr class={selected === row.id ? 'danger' : ''}
        ><td class="col-md-1">{row.id}</td><td class="col-md-4"
```

```
<For each={data()}>
  {(row) => {
```

Create (many) rows

SVELTE • TEMPLATE SYNTAX

{#each ...}

SEE ALSO

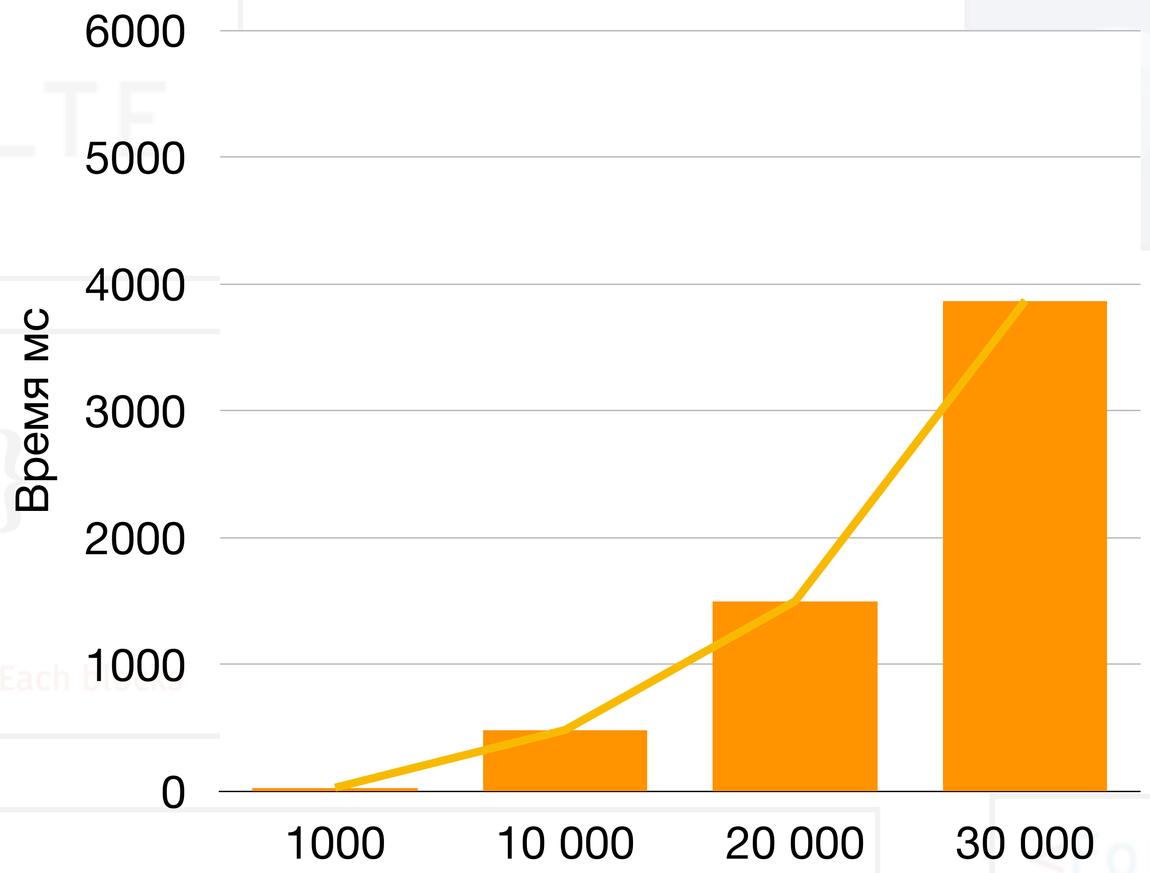
Tutorial • Basic Svelte • Logic • Each blocks

<For each={...} />

```
{#each data as row (row)}
    <tr class={selected === row.id ? 'danger' : ''}
        ><td class="col-md-1">{row.id}</td><td class="col-md-4"
```

```
<For each={data()}>
  {(row) => {
```

Create (many) rows



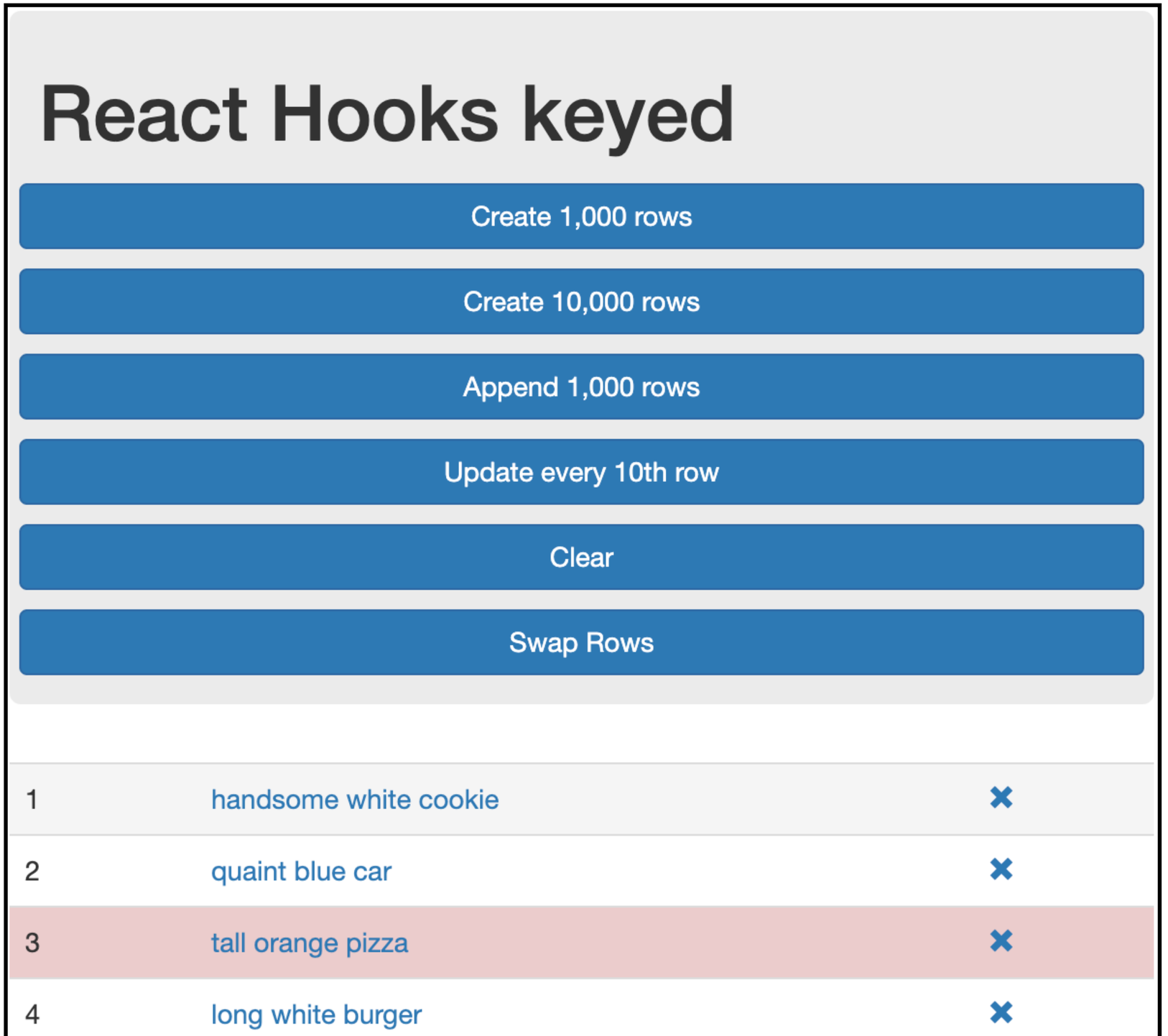
性能

Performance

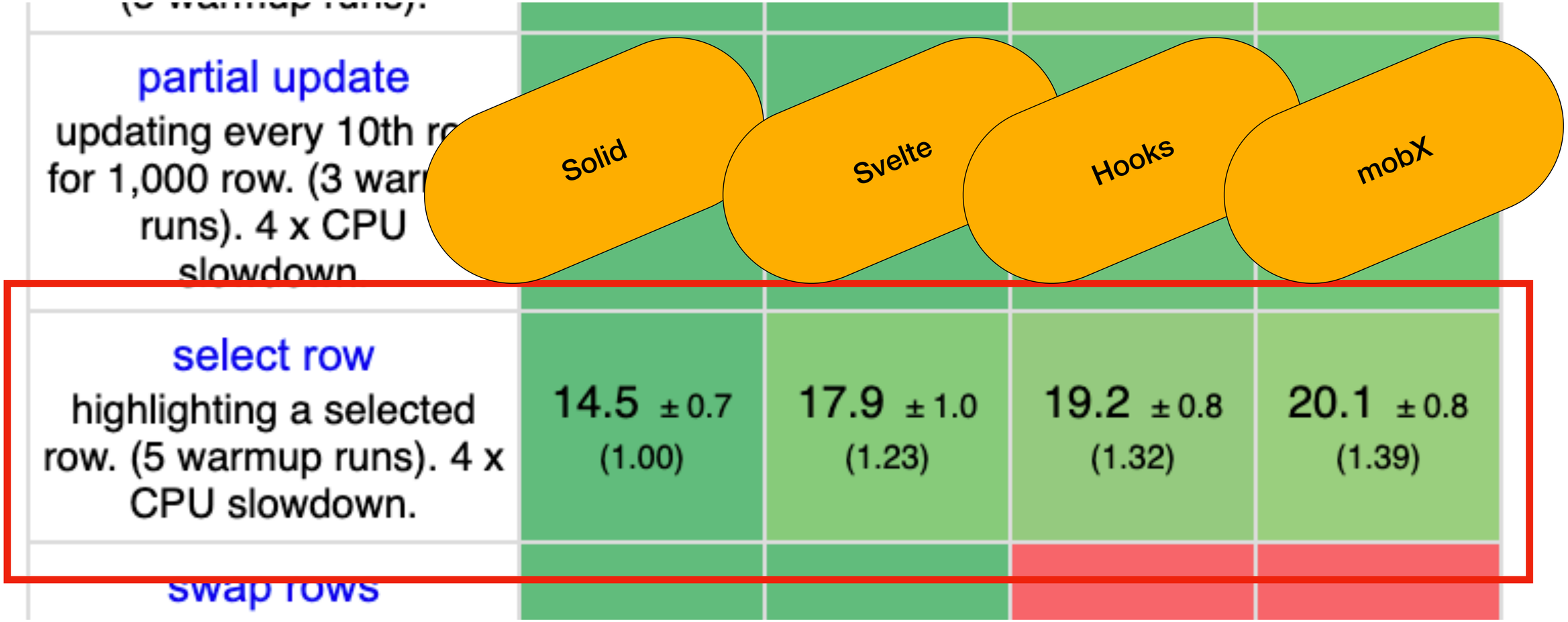
Longer is better!

* The results are based on js-framework-benchmark, and tested on a MacBook Pro M1 Max.

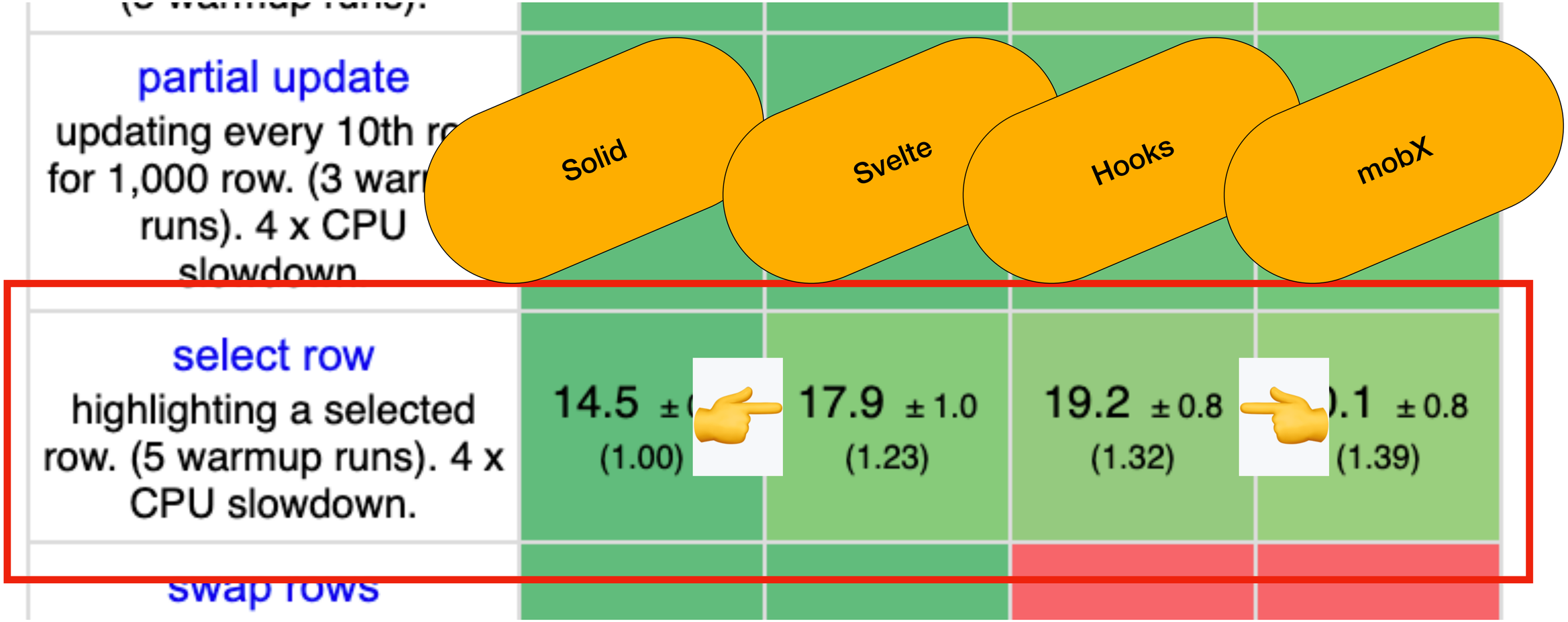
Select row


React Hooks keyed
Create 1,000 rows
Create 10,000 rows
Append 1,000 rows
Update every 10th row
Clear
Swap Rows
1 handsome white cookie
2 quaint blue car
3 tall orange pizza
4 long white burger

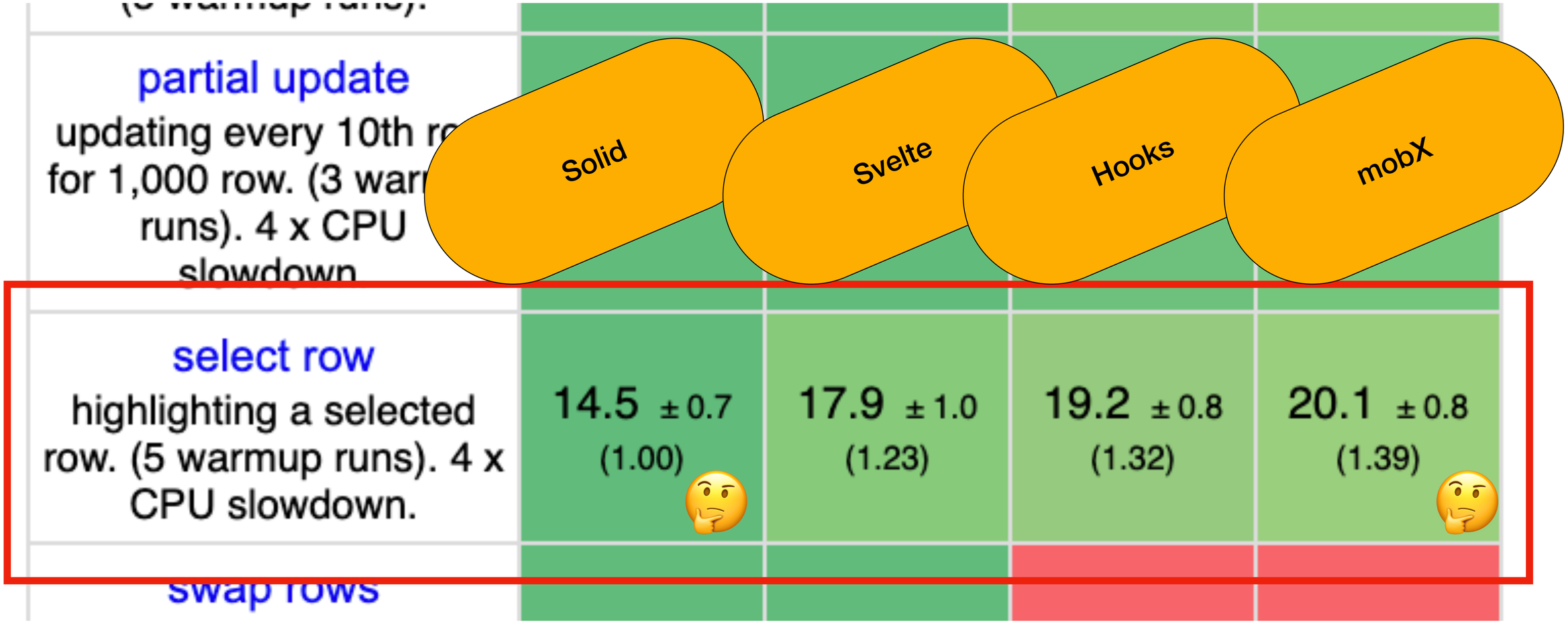
Select row


partial update
updating every 10th r
for 1,000 row. (3 war
runs). 4 x CPU
slowdown
Solid
Svelte
Hooks
mobX
select row
highlighting a selected row. (5 warmup runs). 4 x CPU slowdown.
14.5 ± 0.7
(1.00)
17.9 ± 1.0
(1.23)
19.2 ± 0.8
(1.32)
20.1 ± 0.8
(1.39)
swap rows

Select row


partial update
updating every 10th r
for 1,000 row. (3 war
runs). 4 x CPU
slowdown
Solid
Svelte
Hooks
mobX
select row
highlighting a selected
row. (5 warmup runs). 4 x
CPU slowdown.
14.5 ±
(1.00)
17.9 ±1.0
(1.23)
19.2 ±0.8
(1.32)
.1 ±0.8
(1.39)
swap rows

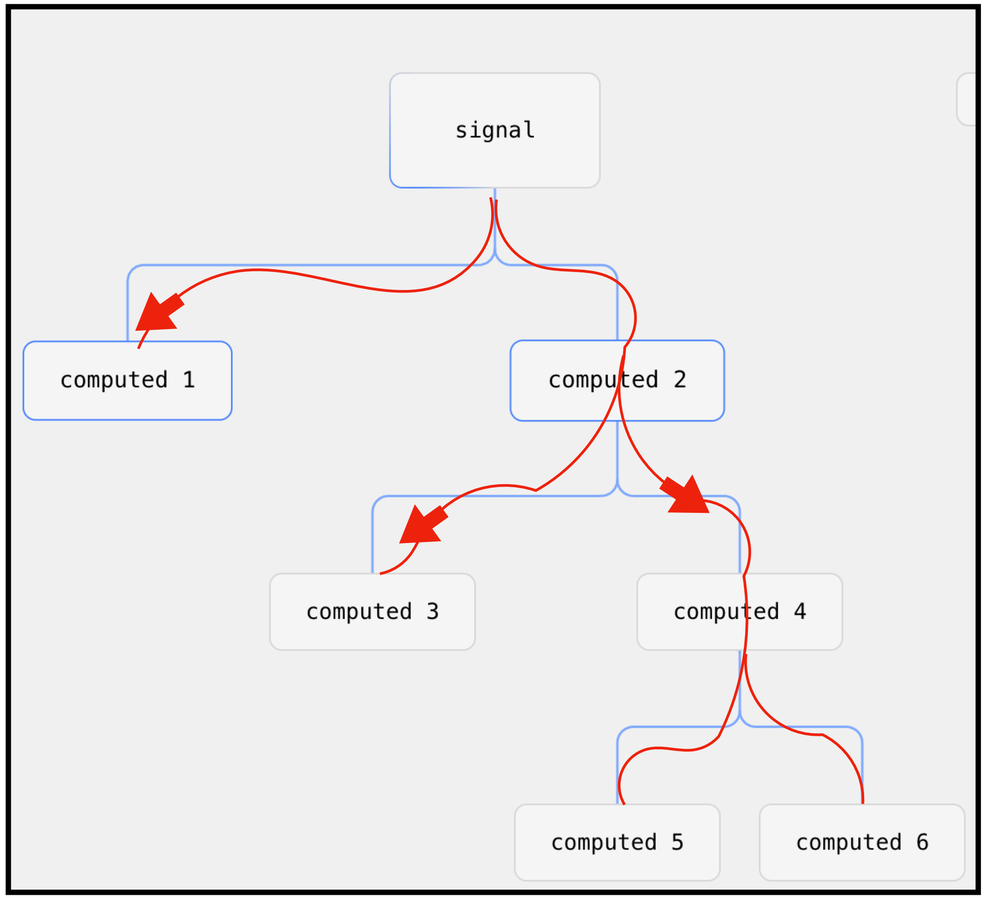
Select row


partial update
updating every 10th ro
for 1,000 row. (3 warr
runs). 4 x CPU
slowdown
Solid
Svelte
Hooks
mobX
select row
highlighting a selected row. (5 warmup runs). 4 x CPU slowdown.
14.5 ±0.7
(1.00)
17.9 ±1.0
(1.23)
19.2 ±0.8
(1.32)
20.1 ±0.8
(1.39)
swap rows

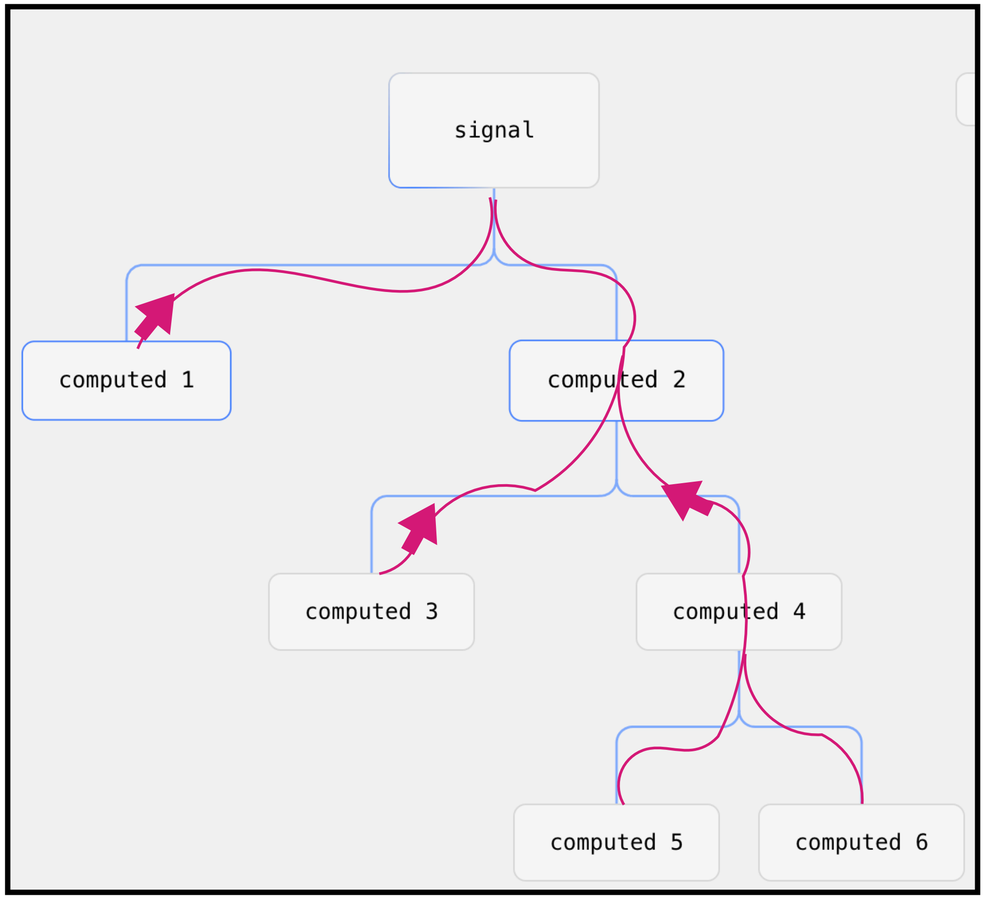
Реактивность в React vs сигналы

Реактивность сигналов

Write (push-based)

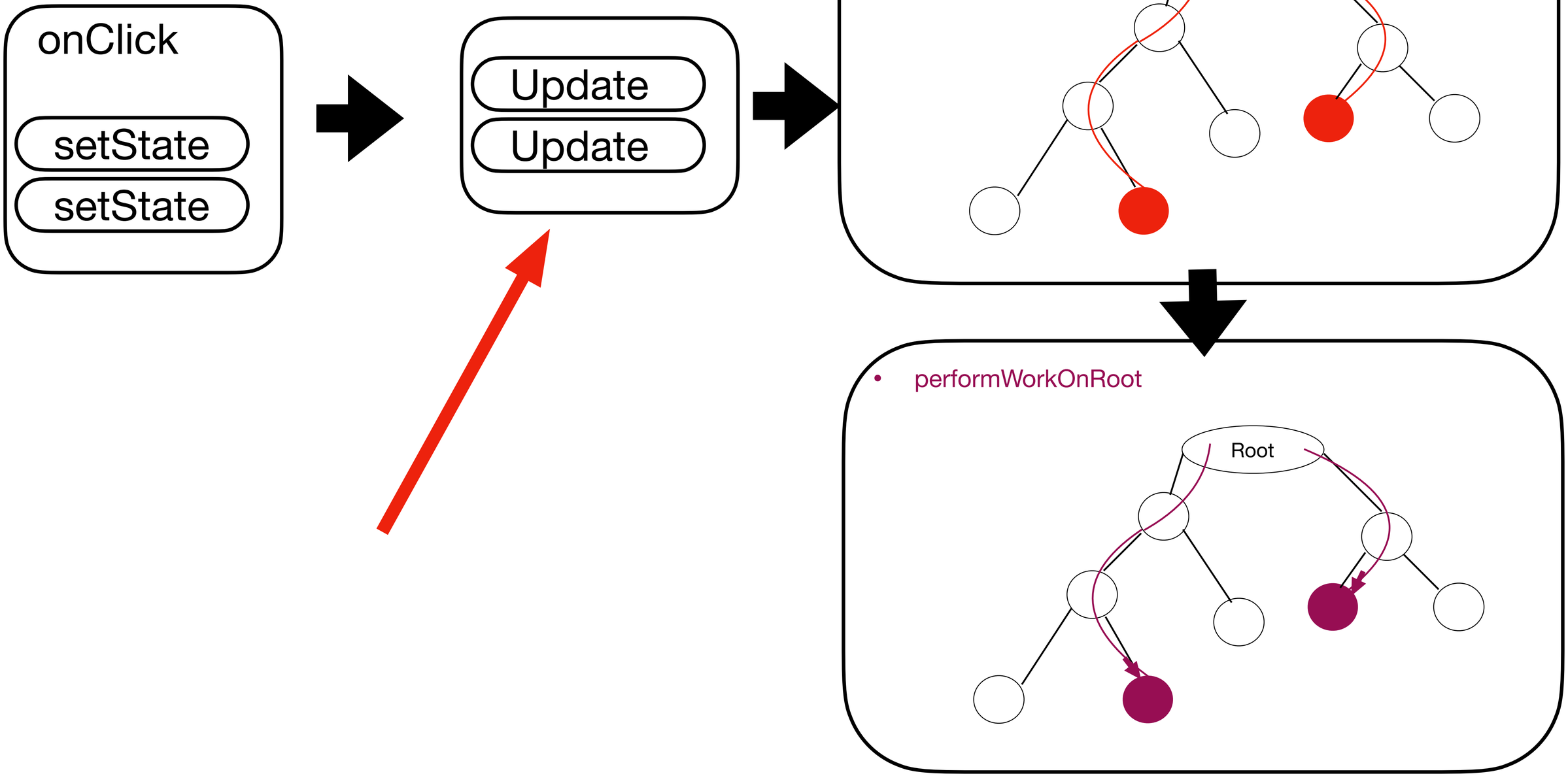


Read (pull-based)



https://willybrauner.com/journal/signal-the-push-pull-based-algorithm

Реактивность в react



Реактивность в react

onClick

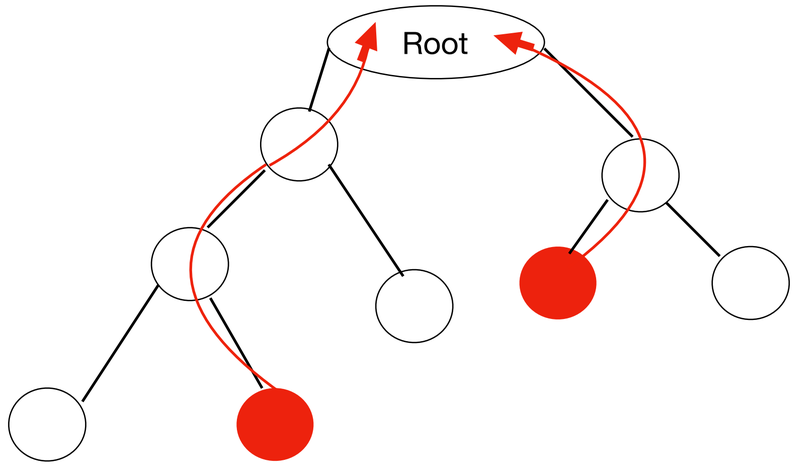
setState

setState

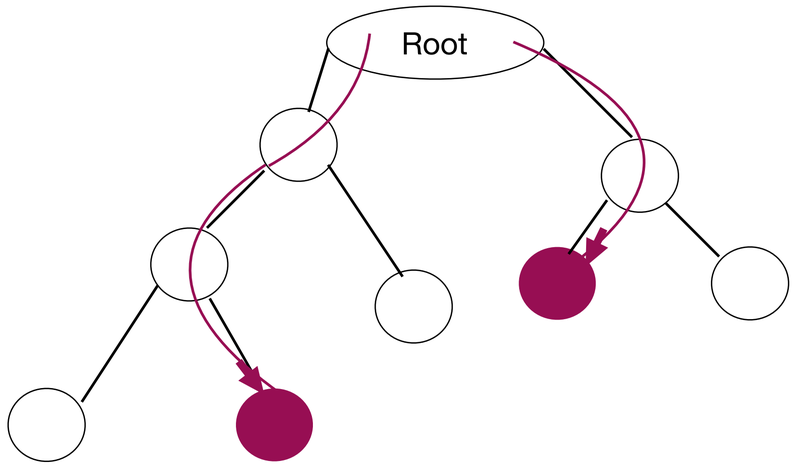
Update

Update

- markUpdateLaneFromFiberToRoot (lanes/childLanes)

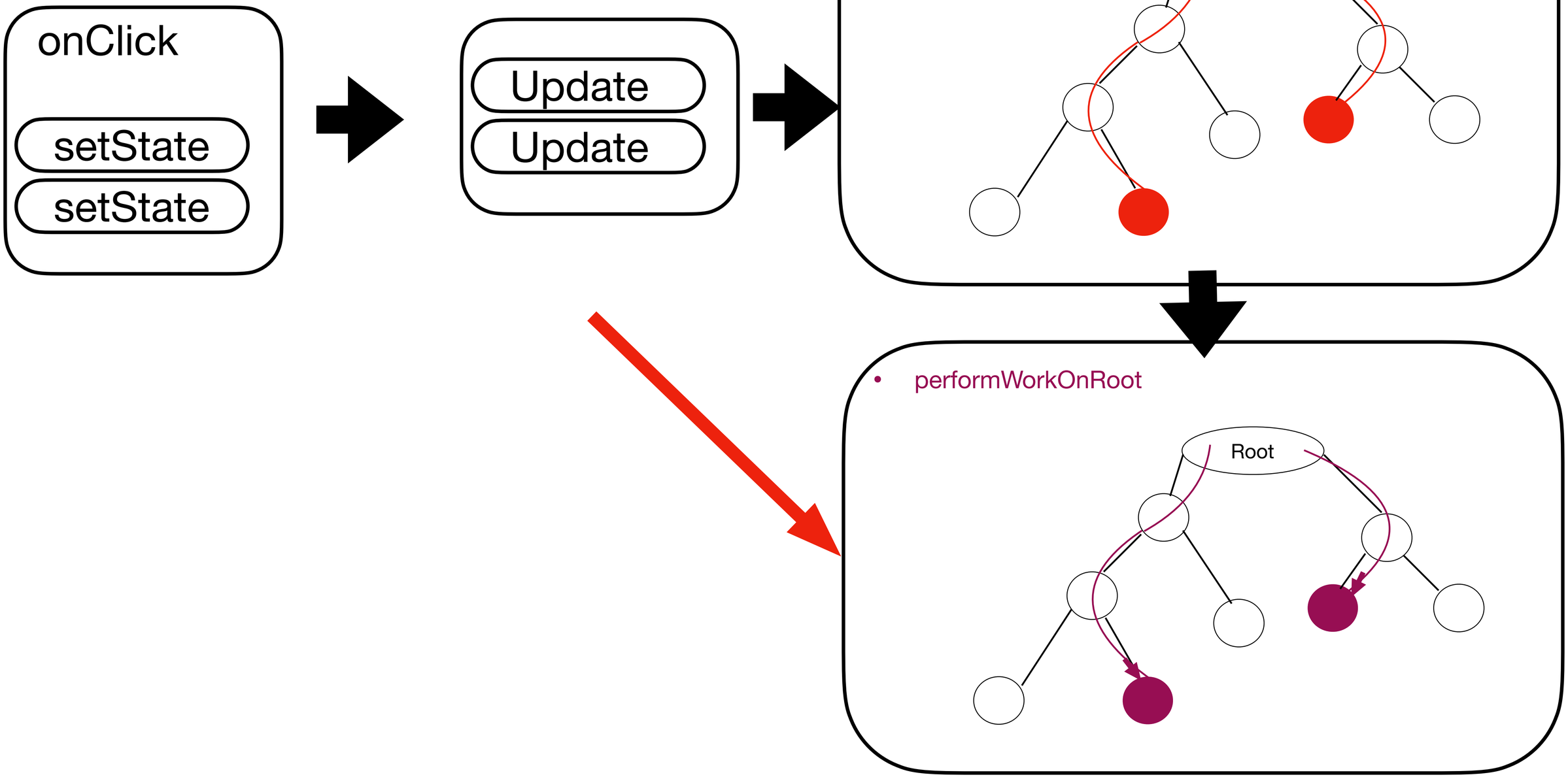


- performWorkOnRoot



lanes/childLanes

Реактивность в react



Сигналы в Angular

Signal API

Сигналы в Angular

Поэтому разработчики Angular ввели Signal API — `Local Change Detection`. Теперь сам компонент уведомляет, что в нем что-то изменилось, и не делает дополнительных проверок по всему дереву компонентов. Мы просто ищем нужный компонент, который имеет определенный флаг.

https://habr.com/ru/companies/tbank/articles/928206/

Сигналы в Angular

В игру вступают новые флаги: RefreshView и HasChildViewsToRefresh. Компонент, в котором изменился сигнал, помечается флагом `RefreshView`, а все его предки — флагом `HasChildViewsToRefresh`. В таком случае, пока мы не дойдем до `RefreshView`-компонента, все остальные будут пропущены и лишь он будет проверен и обновлен.

https://habr.com/ru/companies/tbank/articles/928206/

Сигналы в Angular

В игру вступают новые флаги: RefreshView и HasChildViewsToRefresh. Компонент, в котором изменился сигнал, помечается флагом `RefreshView`, а все его предки — флагом `HasChildViewsToRefresh`. В таком случае, пока мы не дойдем до `RefreshView`-компонента, все остальные будут пропущены и лишь он будет проверен и обновлен.

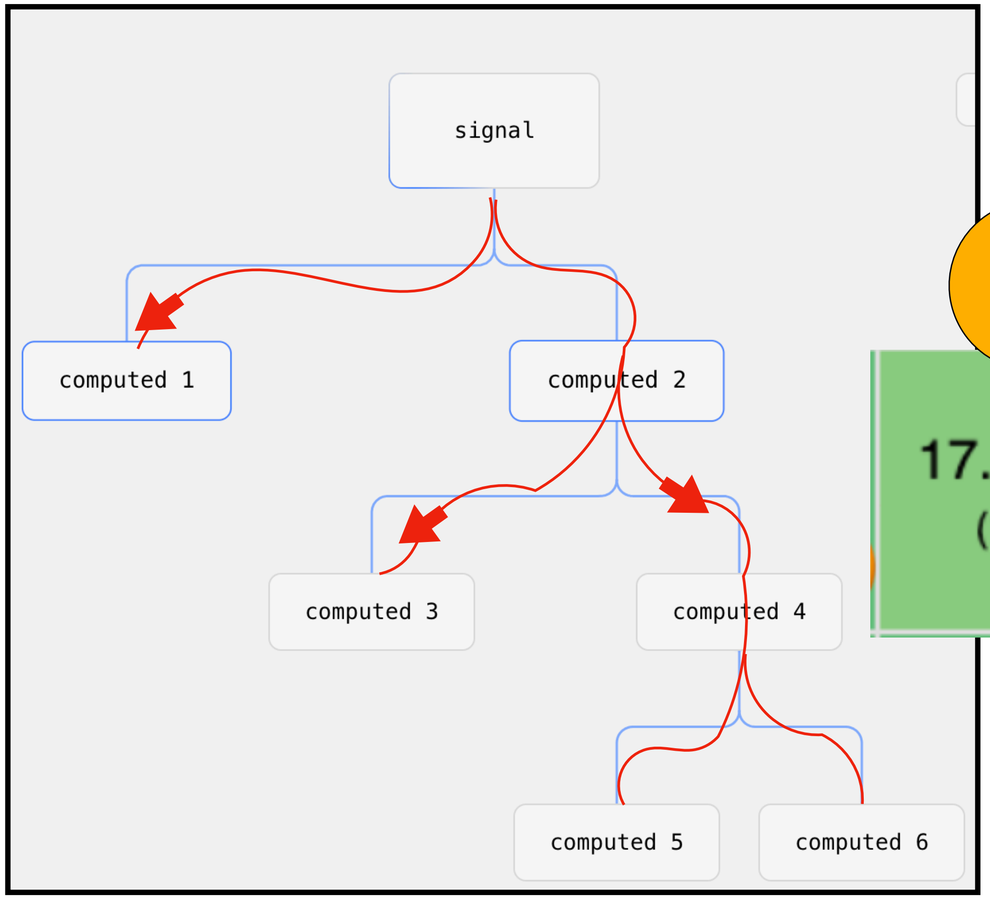
https://habr.com/ru/companies/tbank/articles/928206/

Сигналы в React?

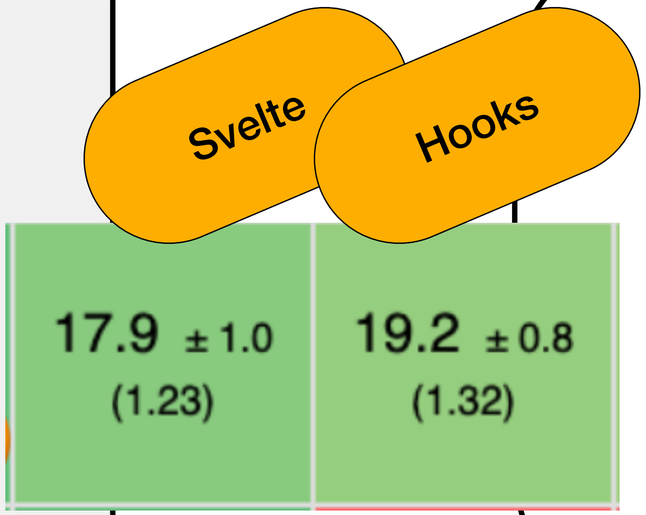
В игру вступают новые флаги: lanes и childLanes.
Компонент, в котором изменился сигнал, помечается флагом lanes, а все его предки — флагом childLanes. В таком случае, пока мы не дойдем до lanes-компонента, все остальные будут пропущены и лишь он будет проверен и обновлен.

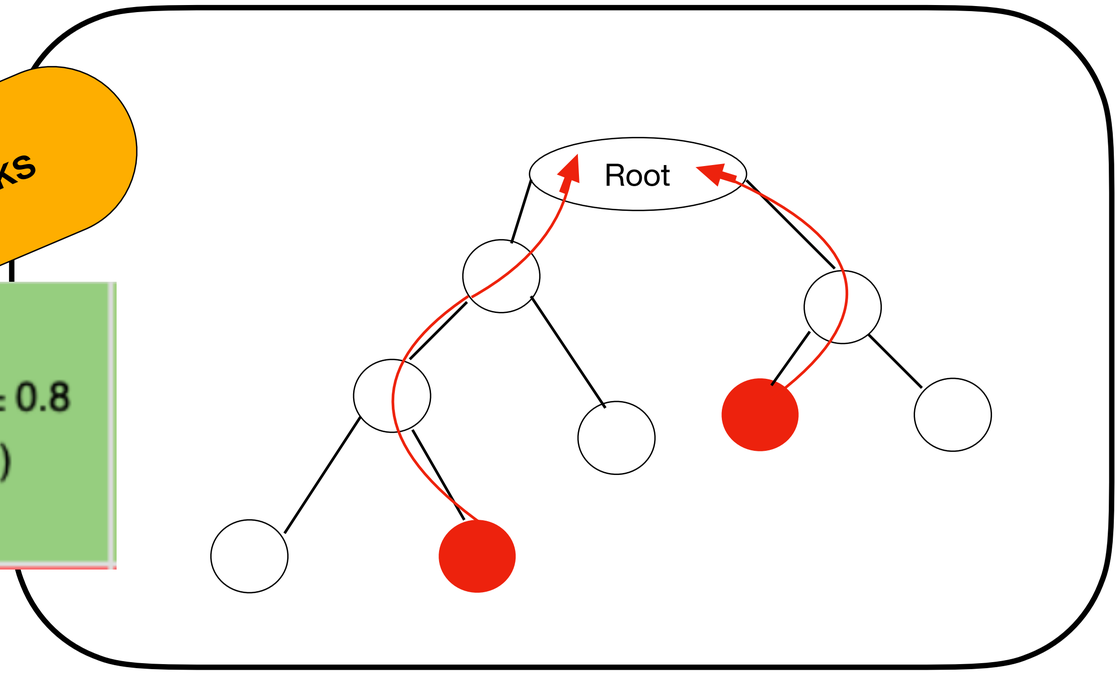
https://habr.com/ru/companies/tbank/articles/928206/

Сравнение реактивности



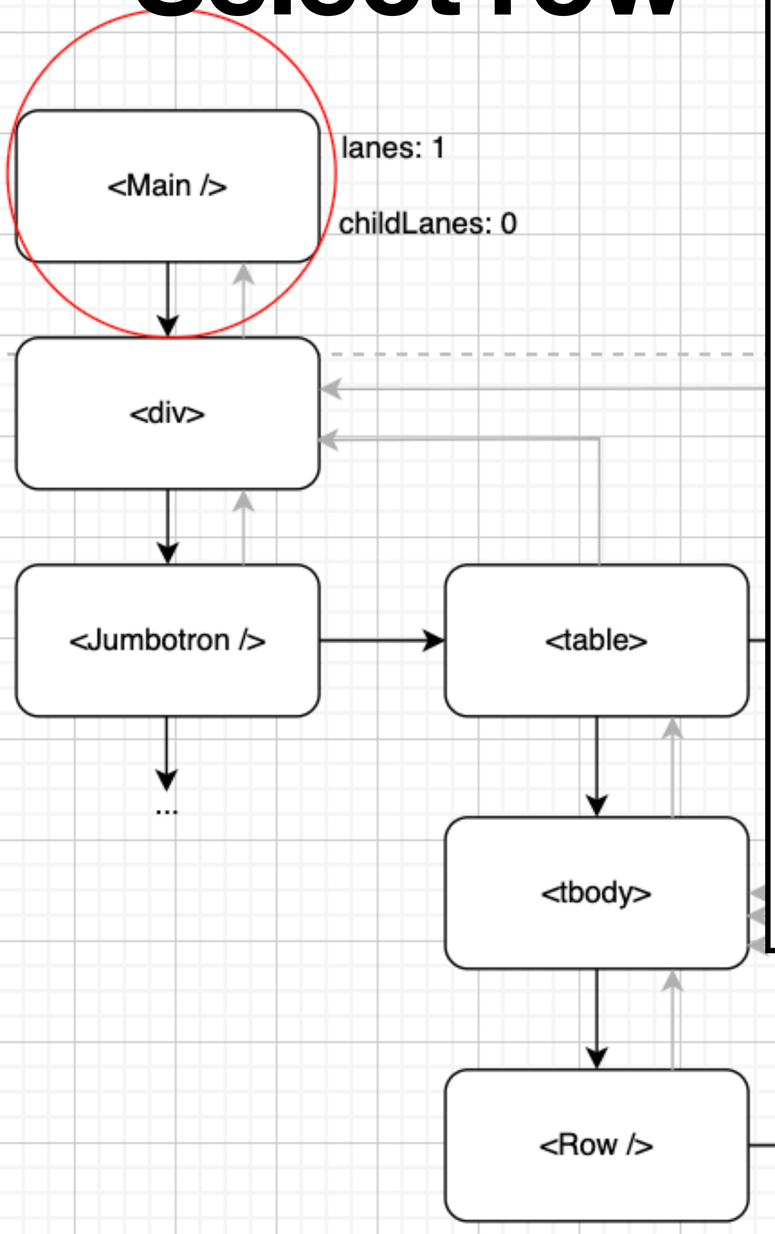
Signal





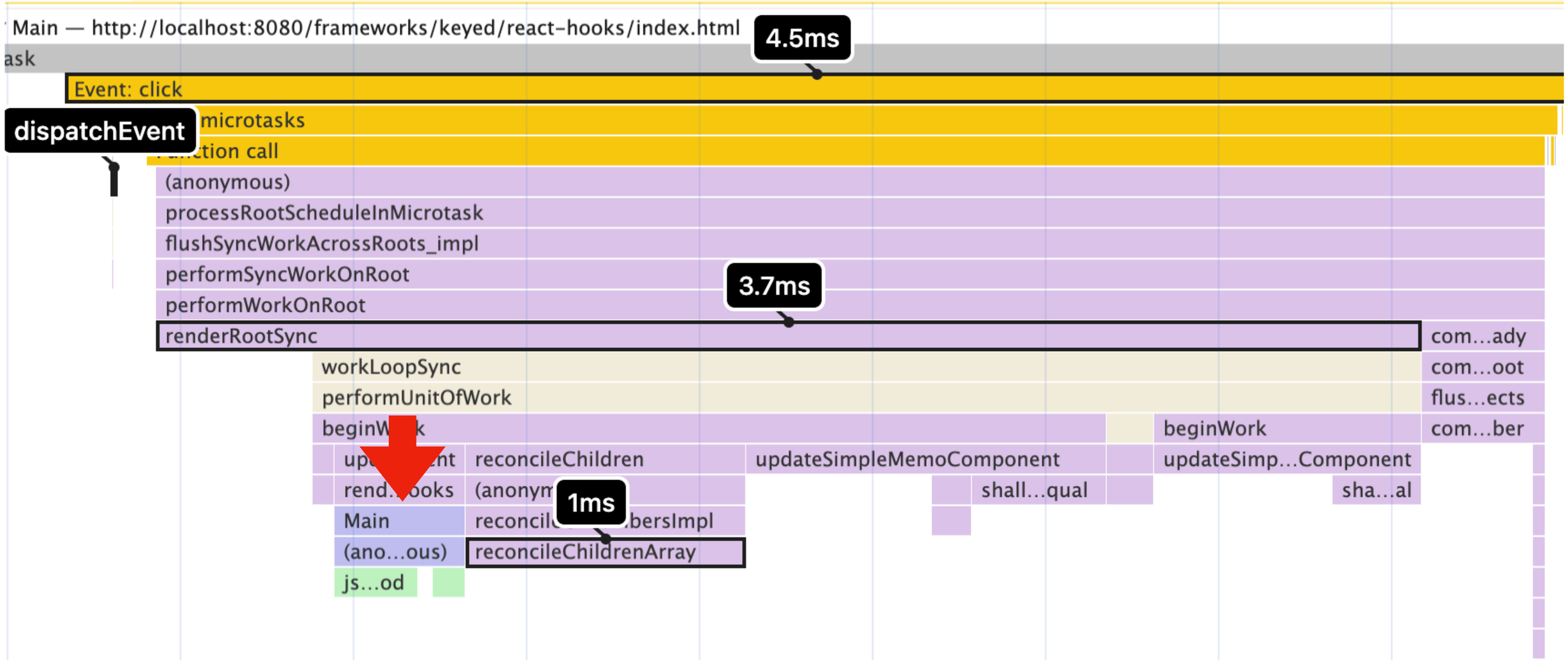
React

Select row




```jsx
const Main = () => {
  const [{ data, selected }, dispatch] = useReducer(listReducer, initialState);

  return (<div className="container">
    <Jumbotron dispatch={dispatch} />
    <table className="table table-hover table-striped test-data">
    {!!data.length &&
      <tbody>
      {data.map(item => (
        <Row
          key={item.id}
          item={item}
          selected={selected === item.id}
          dispatch={dispatch} />
      ))}
      </tbody>
    }
    </table>
    <span className="preloadicon glyphicon glyphicon-remove" aria-hidden="true" />
  </div>);
}
```

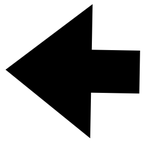
Select row


Main — http://localhost:8080/frameworks/keyed/react-hooks/index.html
4.5ms
Event: click
dispatchEvent
microtasks
(anonymous)
processRootScheduleInMicrotask
flushSyncWorkAcrossRoots_impl
performSyncWorkOnRoot
3.7ms
performWorkOnRoot
renderRootSync
com…ady
workLoopSync
com…oot
performUnitOfWork
flus…ects
beginWork
com…ber
reconcileChildren
updateSimpleMemoComponent
updateSimp…Component
shall…qual
sha…al
1ms
Main
(ano…ous)
reconcileChildrenArray
js…od

Select row

- Использовать внешний стор

```
const Main = () => {
  const data = useSyncExternalStore(store.subscribe, store.getSnapshot);
  const dispatch = store.dispatch;

  return (<div className="container">
    <Jumbotron dispatch={dispatch} />
    <table className="table table-hover table-striped test-data">
    {!!data.length &&
      <tbody>
        {data.map(item => (
          <Row
            key={item.id}
            item={item}
            // selected={selected === item.id}
            dispatch={dispatch} />
        ))}
      </tbody>
    }
    </table>
    <span className="preloadicon glyphicon glyphicon-remove" aria-hidden="true" />
  </div>);
}
```



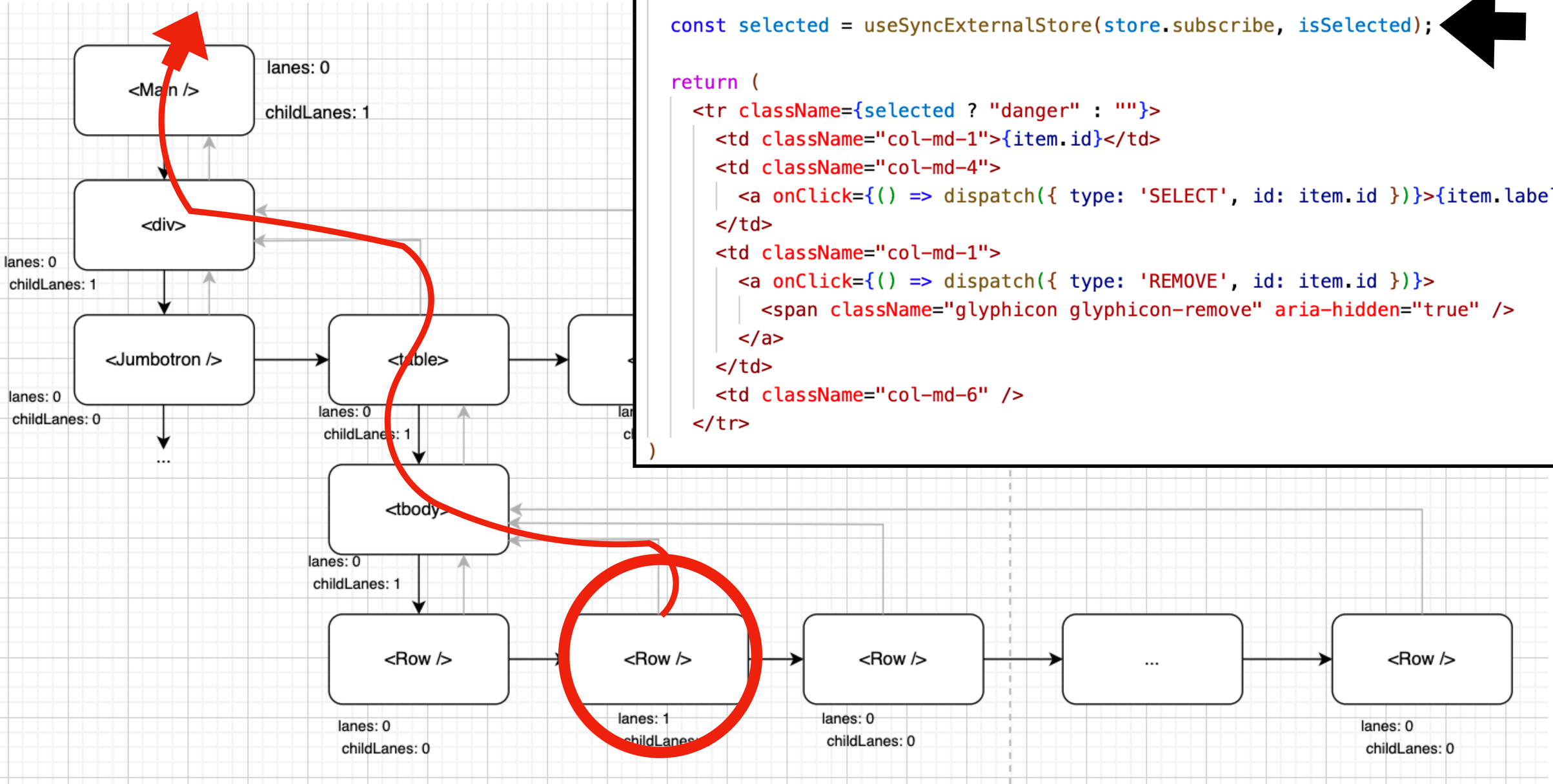
Select row

```
const Main = () => {
  const data = useSyncExternalStor
  const dispatch = store.dispatch;

  return (<div className="containe
    <Jumbotron dispatch={dispatch}
    <table className="table table-
    {!!data.length &&
      <tbody>
        {data.map(item => (
          <Row
            key={item.id}
            item={item}
            // selected={selected
            dispatch={dispatch} />
        ))}
      </tbody>
    }
    </table>
    <span className="preloadicon glyphicon glyphicon-remove" aria-hidden="true" />
  </div>);
}
```

```
const Row = memo(({ item, dispatch }) => {
  const isSelected = useCallback(() => {
    return store.getSnapshot().selected === item.id;
  }, [item.id])

  const selected = useSyncExternalStore(store.subscribe, isSelected);

  return (
    <tr className={selected ? "danger" : ""}>
      <td className="col-md-1">{item.id}</td>
      <td className="col-md-4">
        <a onClick={() => dispatch({ type: 'SELECT', id: item.id })}>{item.label}</a>
      </td>
      <td className="col-md-1">
        <a onClick={() => dispatch({ type: 'REMOVE', id: item.id })}>
          <span className="glyphicon glyphicon-remove" aria-hidden="true" />
        </a>
      </td>
      <td className="col-md-6" />
    </tr>
)
```

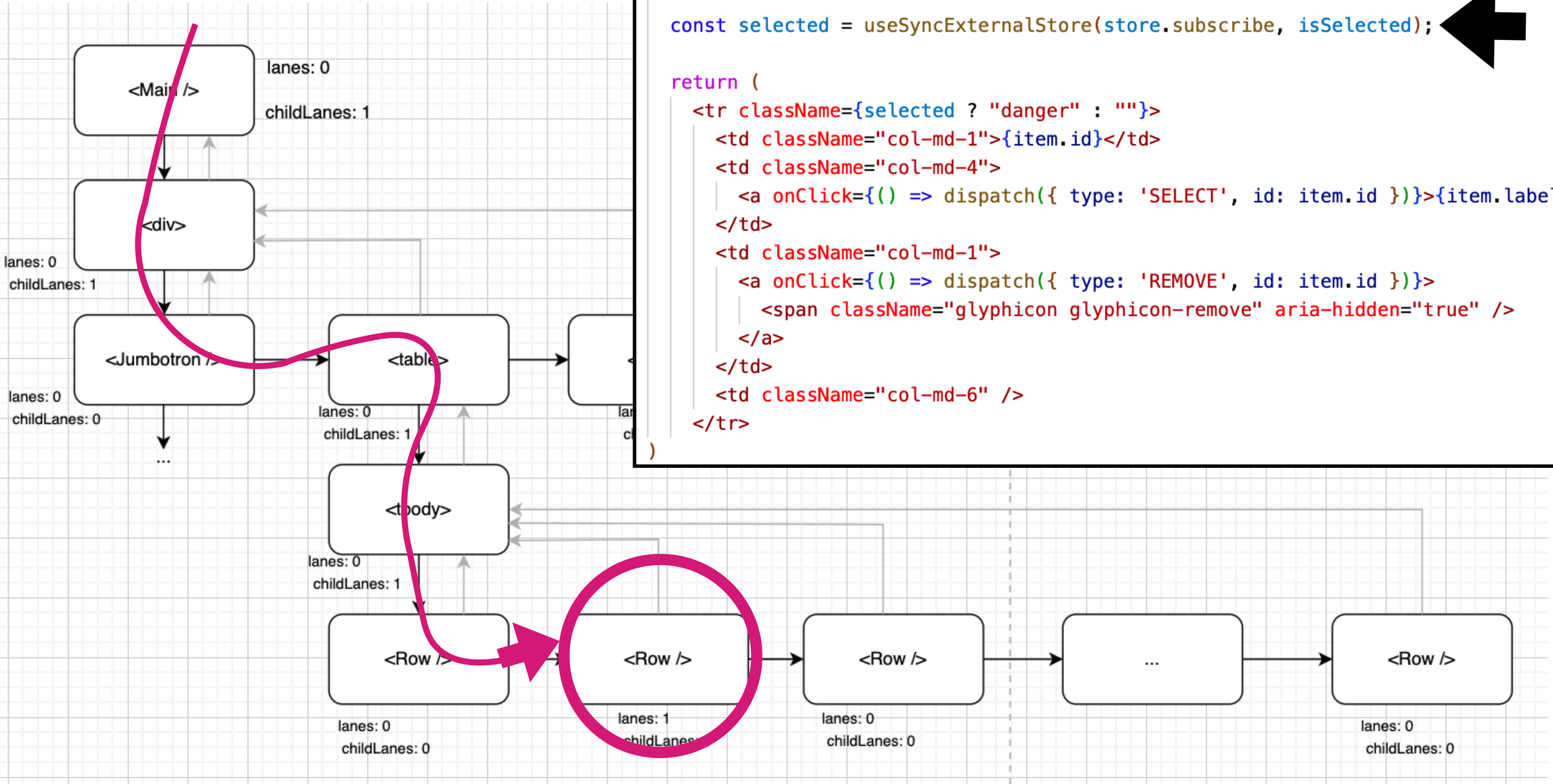
Select row

```
const Row = memo(({ item, dispatch }) => {
  const isSelected = useCallback(() => {
    return store.getSnapshot().selected === item.id;
  }, [item.id])

  const selected = useSyncExternalStore(store.subscribe, isSelected);

  return (
    <tr className={selected ? "danger" : ""}>
      <td className="col-md-1">{item.id}</td>
      <td className="col-md-4">
        <a onClick={() => dispatch({ type: 'SELECT', id: item.id })}>{item.label}</a>
      </td>
      <td className="col-md-1">
        <a onClick={() => dispatch({ type: 'REMOVE', id: item.id })}>
          <span className="glyphicon glyphicon-remove" aria-hidden="true" />
        </a>
      </td>
      <td className="col-md-6" />
    </tr>
)
```



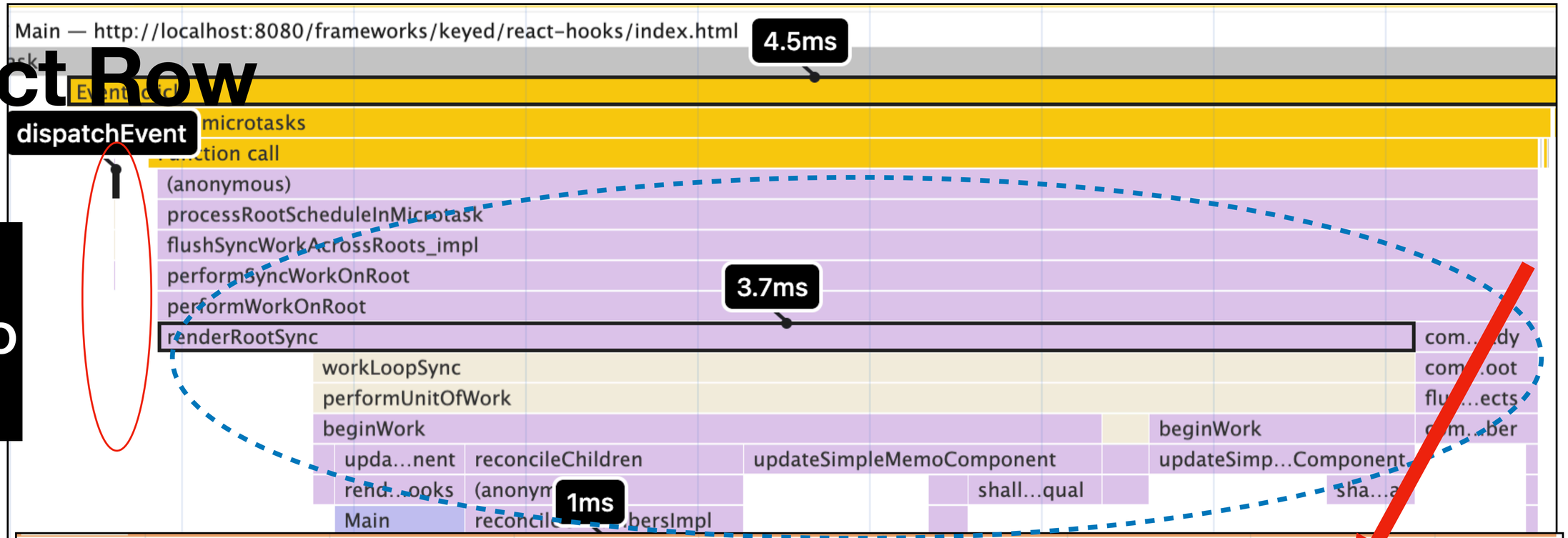
Select row

```jsx
const Row = memo(({ item, dispatch }) => {
  const isSelected = useCallback(() => {
    return store.getSnapshot().selected === item.id;
  }, [item.id])

  const selected = useSyncExternalStore(store.subscribe, isSelected);

  return (
    <tr className={selected ? "danger" : ""}>
      <td className="col-md-1">{item.id}</td>
      <td className="col-md-4">
        <a onClick={() => dispatch({ type: 'SELECT', id: item.id })}>{item.label}</a>
      </td>
      <td className="col-md-1">
        <a onClick={() => dispatch({ type: 'REMOVE', id: item.id })}>
          <span className="glyphicon glyphicon-remove" aria-hidden="true" />
        </a>
      </td>
      <td className="col-md-6" />
    </tr>
)
```

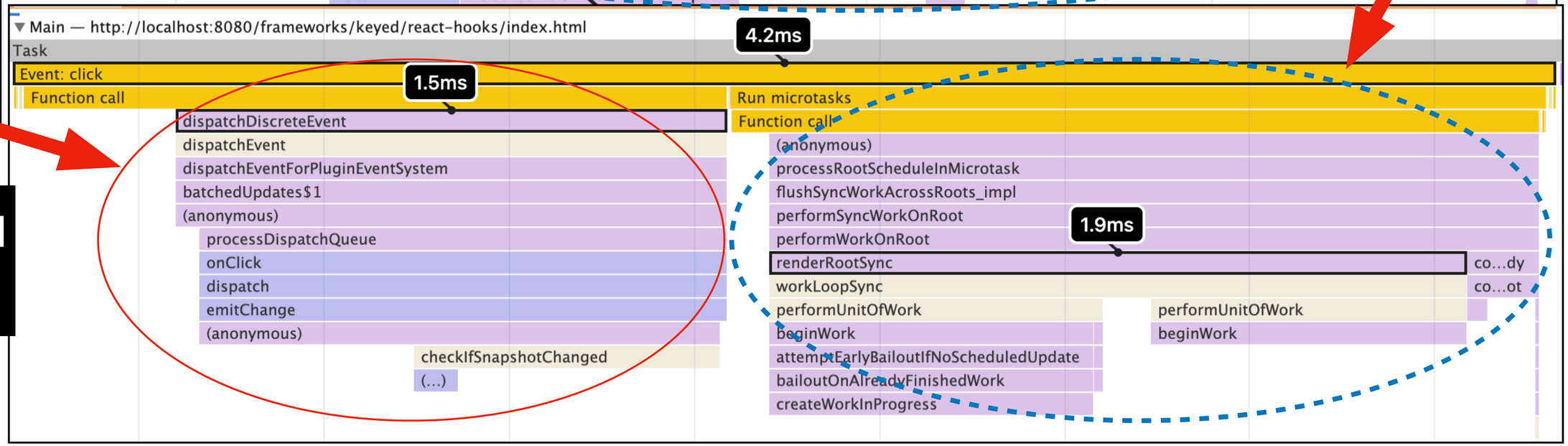


Select Row

Без внешнего Store



Внешний store



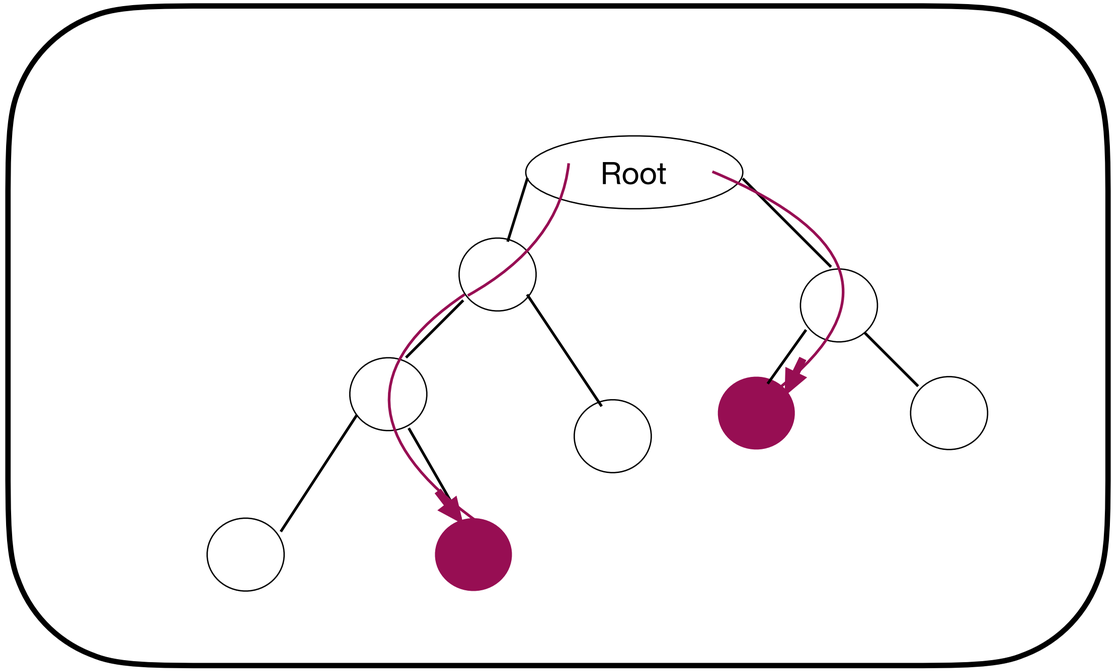
Select row

const Row = memo(({ item, dispatch }) => {
const isSelected = useCallback(() => {
return store.getSnapshot().selected === item.id;
}, [item.id])
const selected = useSyncExternalStore(store.subscribe, isSelected);
return (
Svelte
Hooks
<Main />
lanes: 0
childLanes: 1
<div>
lanes: 0
childLanes: 1
<Jumbotron />
lanes: 0
childLanes: 0
...
<table>
lanes: 0
childLanes: 1
<tbody>
lanes: 0
childLanes: 1
<Row />
lanes: 0
childLanes: 0
<Row />
lanes: 1
<Row />
lanes: 0
childLanes: 0
lanes: 0
childLanes: 0
17.9 ± 1.0
(1.23)
19.2 ± 0.8
18.1 ± 0.6

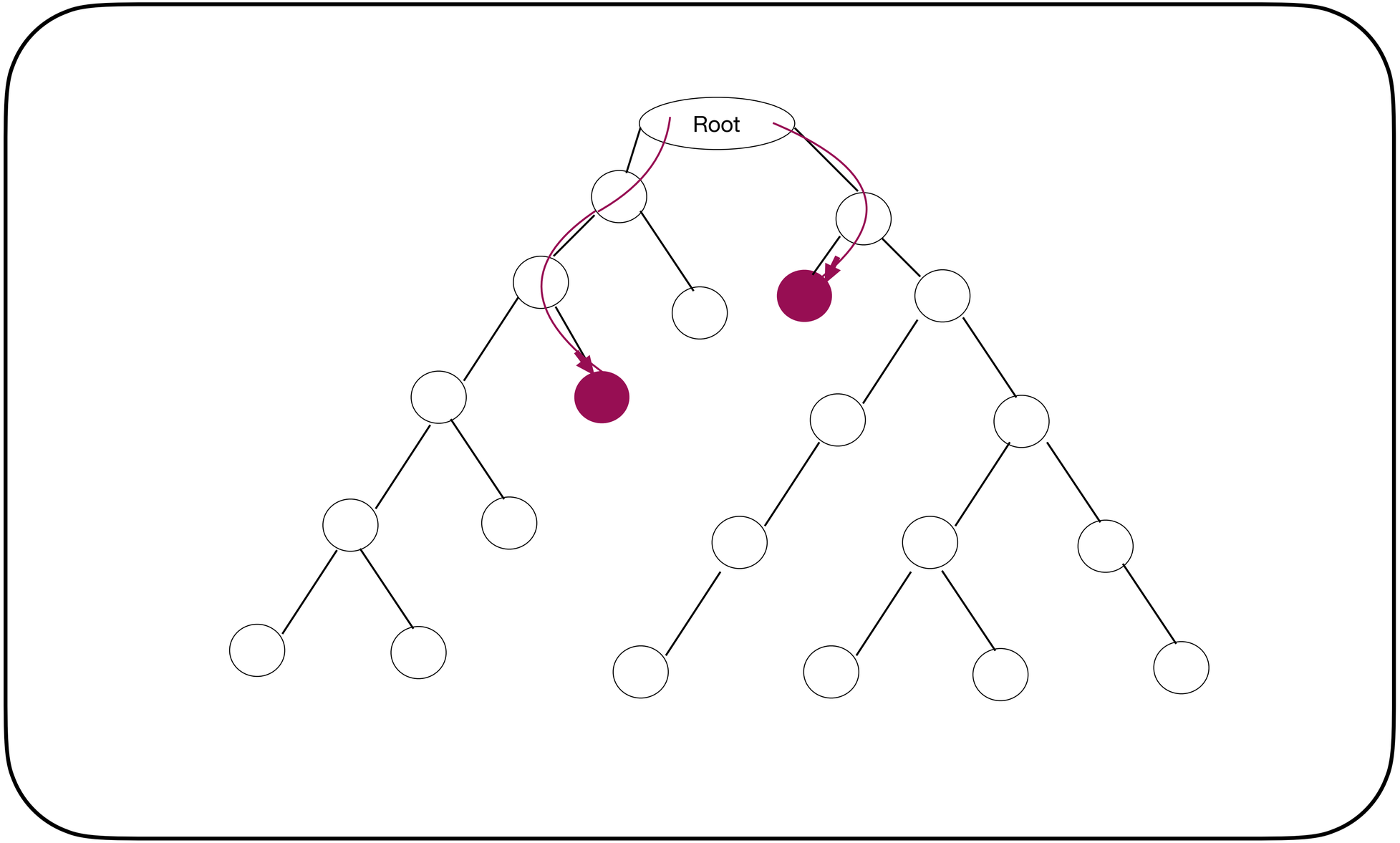
Select row


const Row = memo(({ item, dispatch }) => {
const isSelected = useCallback(() => {
return store.getSnapshot().selected === item.id;
}, [item.id])
const selected = useSyncExternalStore(store.subscribe, isSelected);
return (
Svelte
Hooks
<Main />
lanes: 0
childLanes: 1
<div>
lanes: 0
childLanes: 1
<Jumbotron />
lanes: 0
childLanes: 0
...
<table>
lanes: 0
childLanes: 1
<tbody>
lanes: 0
childLanes: 1
<Row />
lanes: 0
childLanes: 0
<Row />
lanes: 1
<Row />
lanes: 0
childLanes: 0
lanes: 0
childLanes: 0
17.9 ±1.0
(1.23)
19.2 ±0.8
18.1 ±0.6

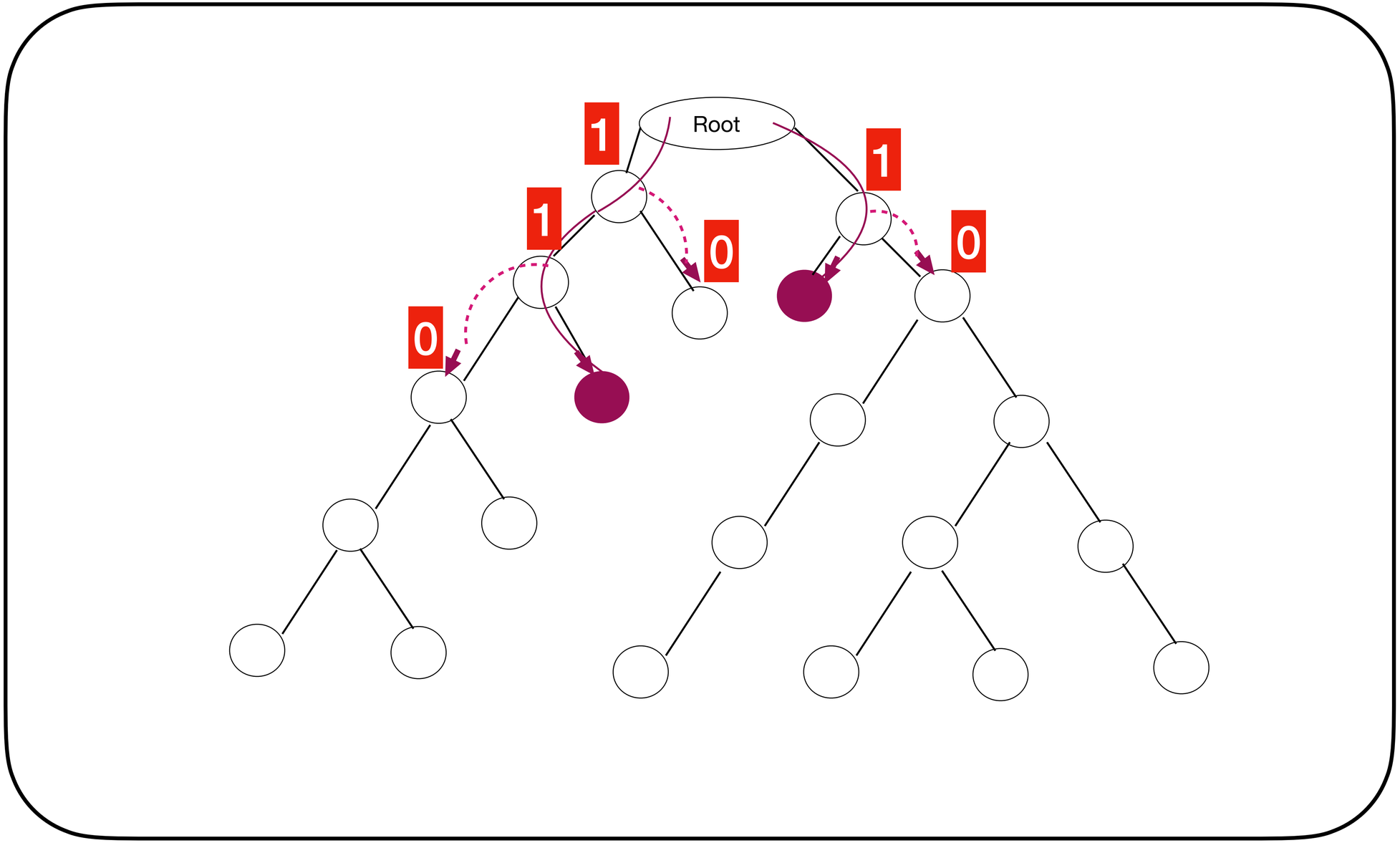
Select row



Select row

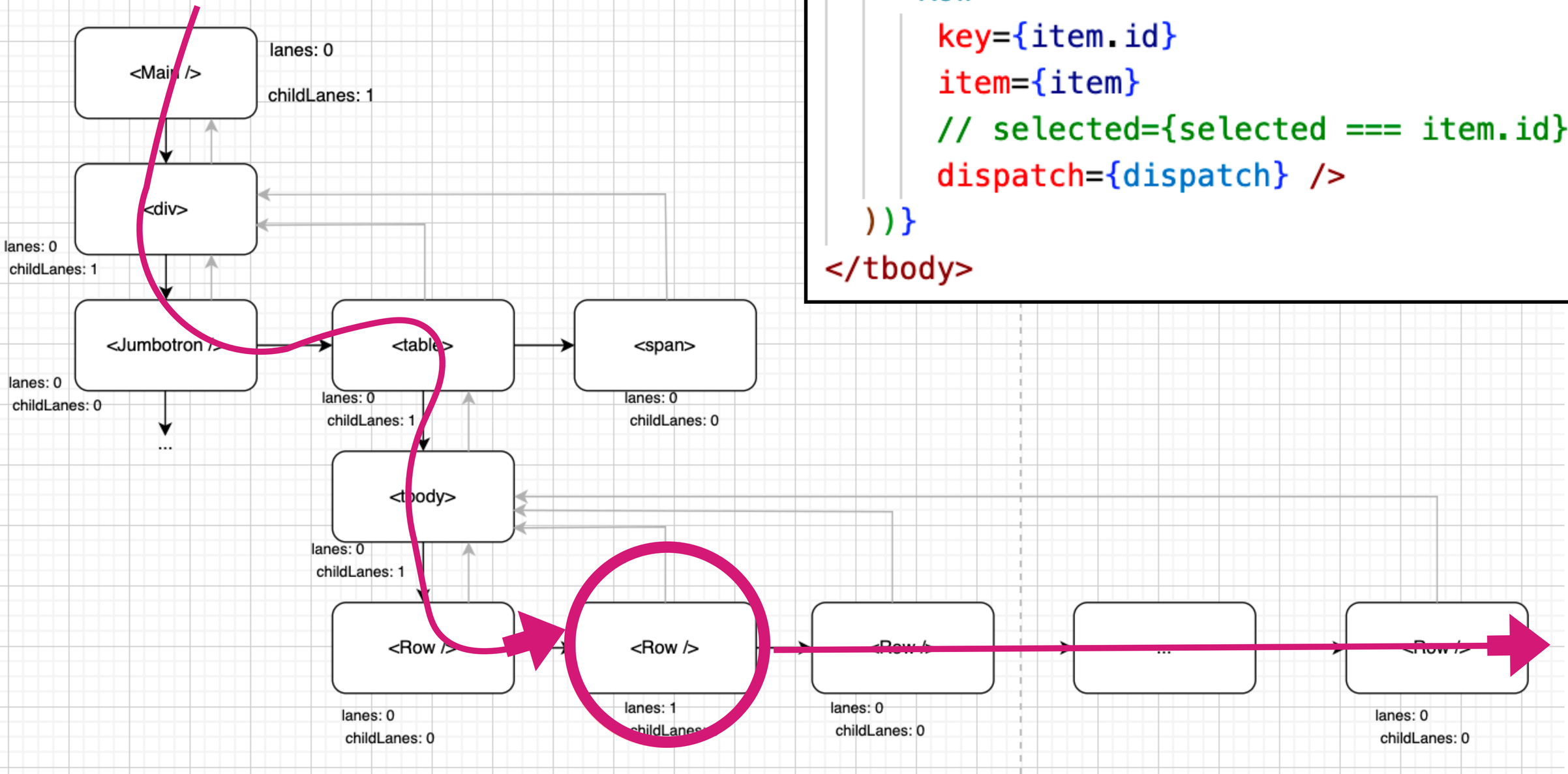


Select row



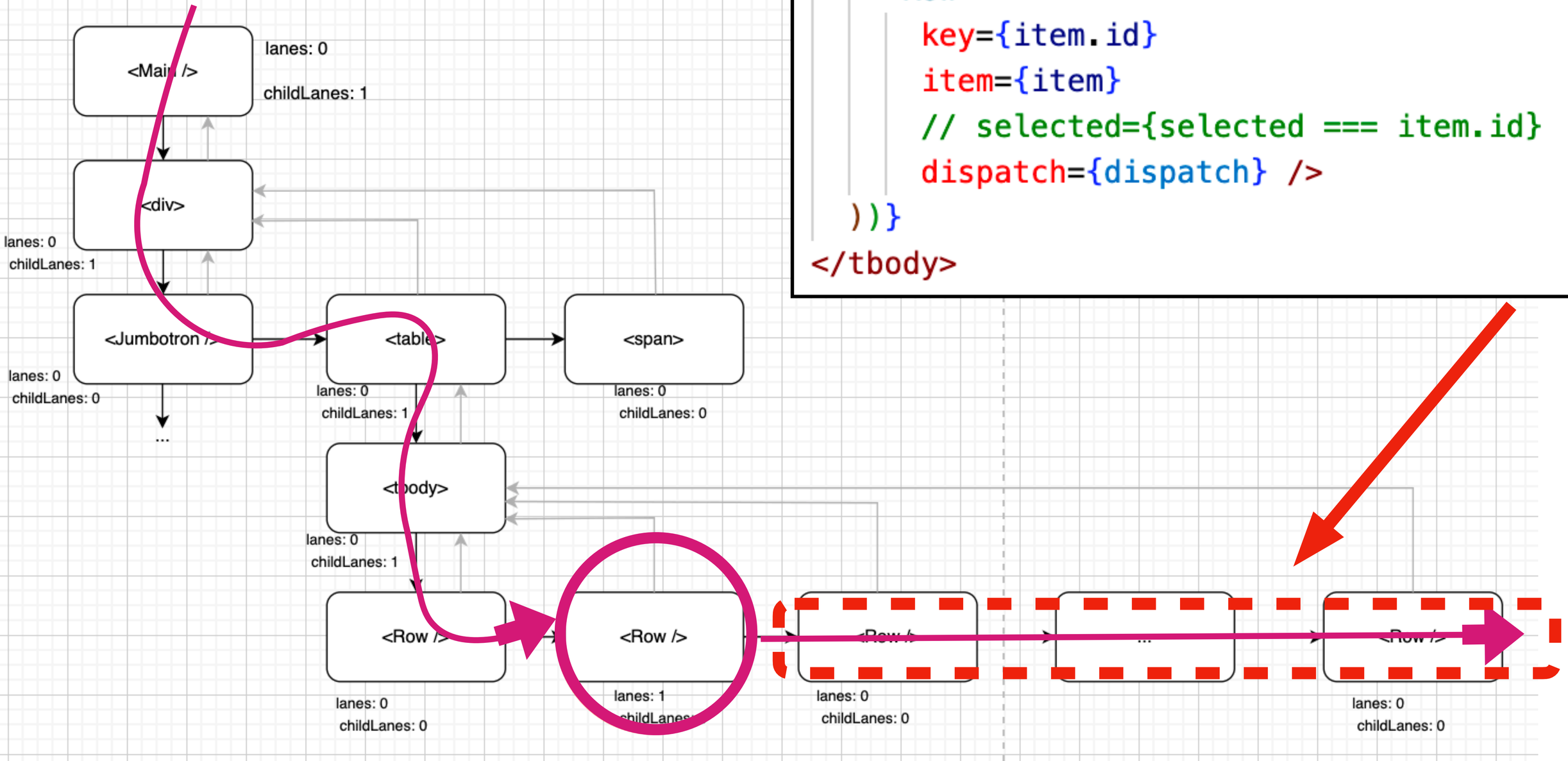
Select row

```
<tbody>
  {data.map(item => (
    <Row
      key={item.id}
      item={item}
      // selected={selected === item.id}
      dispatch={dispatch} />
  ))}
</tbody>
```



Select row

```
<tbody>
  {data.map(item => (
    <Row
      key={item.id}
      item={item}
      // selected={selected === item.id}
      dispatch={dispatch} />
  ))}
</tbody>
```



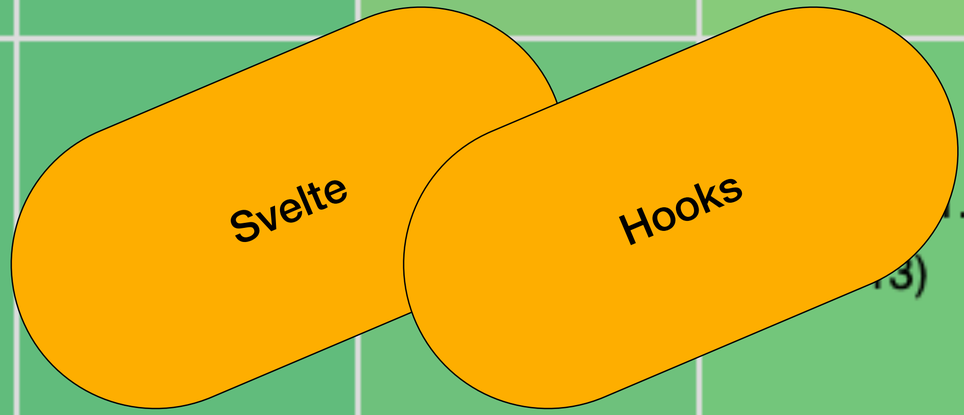
Можно ли ещё сократить render?

```
<tbody>
  <>
    {data.slice(0, 500).map(item => (
      <Row
        key={item.id}
        item={item}
        // selected={selected === item.id}
        dispatch={dispatch} />
    ))}
  </>
  <>
    {data.slice(500).map(item => (
      <Row
        key={item.id}
        item={item}
        // selected={selected === item.id}
        dispatch={dispatch} />
    ))}
  </>
</tbody>
```

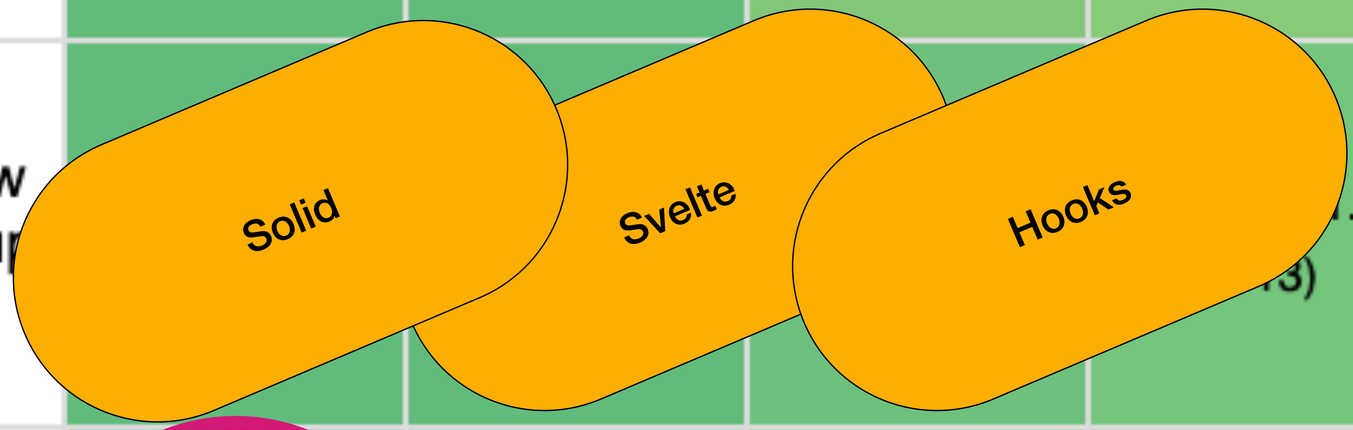
Можно ли ещё сократить render?

O(n) -> O(log n)

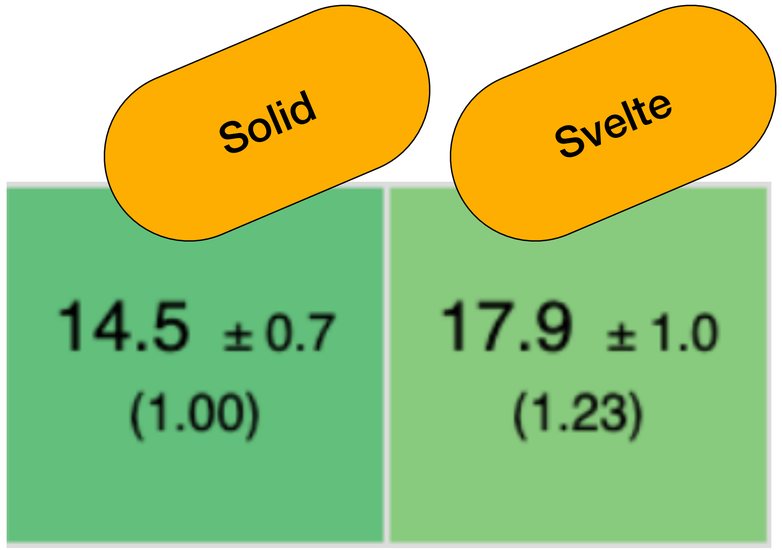
creating 1,000 rows. (5 warmup runs).	90.2 ± 1.1 (1.00)	92.8 ± 1.0 (1.03)	105.3 ± 0.6 (1.17)	114.0 ± 1.0 (1.26)
replace all rows updating all 1,000 rows. (5 warmup runs).	97.9 ± 1.2 (1.00)	99.4 ± 0.9 (1.02)	118.8 ± 0.9 (1.21)	123.9 ± 1.2 (1.27)
partial update updating every 10th row for 1,000 row. (3 warmup runs). 4 x CPU slowdown.	55.6 ± 0.4 (1.00)			
select row highlighting a selected row. (5 warmup runs). 4 x CPU slowdown.	14.5 ± 0.7 (1.00)	17.9 ± 1.0 (1.23)	16.6 ± 0.6 (1.14)	20.1 ± 0.8 (1.39)
swap rows swap 2 rows for table with 1,000 rows. (5 warmup runs). 4 x CPU	65.4 ± 0.9 (1.01)	64.8 ± 1.4 (1.00)	349.7 ± 2.7 (5.40)	342.6 ± 3.3 (5.29)



creating 1,000 rows. (5 warmup runs).	90.2 ± 1.1 (1.00)	92.8 ± 1.0 (1.03)	105.3 ± 0.6 (1.17)	114.0 ± 1.0 (1.26)
replace all rows updating all 1,000 rows. (5 warmup runs).	97.9 ± 1.2 (1.00)	99.4 ± 0.9 (1.02)	118.8 ± 0.9 (1.21)	123.9 ± 1.2 (1.27)
partial update updating every 10th row for 1,000 row. (3 warmup runs). 4 x CPU slowdown.				.0 3)
select row highlighting a selected row. (5 warmup runs). 4 x CPU slowdown.	14.5 ± 0.7 (1.00)	17.9 ± 1.0 (1.23)	16.6 ± 0.6 (1.14)	20.1 ± 0.8 (1.39)
swap rows swap 2 rows for table with 1,000 rows. (5 warmup runs). 4 x CPU	65.4 ± 0.9 (1.01)	64.8 ± 1.4 (1.00)	349.7 ± 2.7 (5.40)	342.6 ± 3.3 (5.29)



Сравнение реактивности



Сравнение реактивности

Solid

Svelte

partial update updating every 10th row for 1,000 row. (3 warmup runs). 4 x CPU slowdown.	55.6 ± 0.4 (1.00)	56.4 ± 0.9 (1.01)

Select row (Svelte)

```
let selected = $state.raw();
```

```
<tbody>
    {#each data as row (row)}
        <tr class={selected === row.id ? 'danger' : ''}
            ><td class="col-md-1">{row.id}</td><td class="col-md-4"
                ><a
                    onclick={() => {
                        selected = row.id;
                    }}>{row.label}</a
                ></td
            ><td class="col-md-1"
                ><a onclick={() => remove(row)}
                    ><span class="glyphicon glyphicon-remove" aria-hidden="true" /></a
                ></td
            ><td class="col-md-6" /></tr
        >
    {/each}
</tbody>
```

Select row (Solid)

```
const [selected, setSelected] = createSignal(null);
```

```
const isSelected = createSelector(selected);
```

```
<tbody>
  <For each={data()}>
    {(row) => {
      let rowId = row.id;
      return (
        <tr class={isSelected(rowId) ? "danger" : ""}>
          <td class="col-md-1" textContent={rowId} />
          <td class="col-md-4">
            <a onClick={() => setSelected(rowId)} textContent={row.label()} />
          </td>
          <td class="col-md-1">
            <a onClick={() => setData((d) => d.toSpliced(d.findIndex((d) => d.id === rowId), 1))}>
              <span class="glyphicon glyphicon-remove" aria-hidden="true" />
            </a>
          </td>
          <td class="col-md-6" />
        </tr>
      ); // prettier-ignore
    }}
  </For>
</tbody>
```

Select row (Solid)

https://docs.solidjs.com/reference/secondary-primitives/create-selector#createselector

```
const [selectedId, setSelectedId] = createSignal()
const isSelected = createSelector(selectedId)

<For each={list()}>
  {(item) => <li classList={{ active: isSelected(item.id) }}>{item.name}</li>}
</For>
```

Select row (Solid)

https://docs.solidjs.com/reference/secondary-primitives/create-selector#createselector

```
const [selectedId, setSelectedId] = createSignal()
const isSelected = createSelector(selectedId)

<For each={list()}>
  {(item) ⇒ <li classList={{ active: isSelected(item.id) }}>{item.name}</li>}
</For>
```

Обзор от ИИ

+3

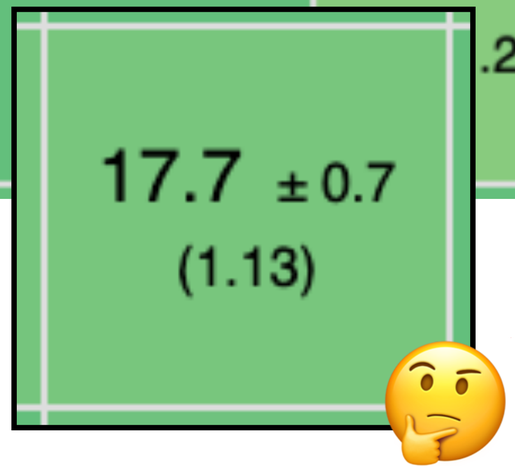
In SolidJS, `createSelector` is a high-performance utility designed to track which item in a list is "selected" without triggering a re-render of every item in that list when the selection changes. SolidJS Docs +2

Select row (Solid без createSelector)

```
<tbody>
  <For each={data()}>
    {(row) => {
      let rowId = row.id;
      return (
        <tr class={isSelected(rowId) ? "danger" : ""}>
        <tr class={selected() === rowId ? "danger" : ""}>
          <td class="col-md-1" textContent={rowId} />
```

Select row (Solid без createSelector)

	solid-v1.9.3	svelte-v5.42.1	react-hooks-v19.2.0
select row highlighting a selected row. (5 warmup runs). 4 x CPU slowdown.	14.5 ± 0.7	17.9 ± 1.0 .23)	16.6 ± 0.6 (1.14)



Select row (S e3 c Selector)

Solid

Svelte

partial update updating every 10th row for 1,000 row. (3 warmup runs). 4 x CPU slowdown.	55.6 ± 0.4 (1.00)	56.4 ± 0.9 (1.01)	react-hooks-v19.2.0
select row highlighting a selected row. (5 warmup runs). 4 x CPU slowdown.	14.5 ± 0.7	17.9 ± 1.0 .23)	16.6 ± 0.6 (1.14)

17.7 ± 0.7
(1.13)

Select row

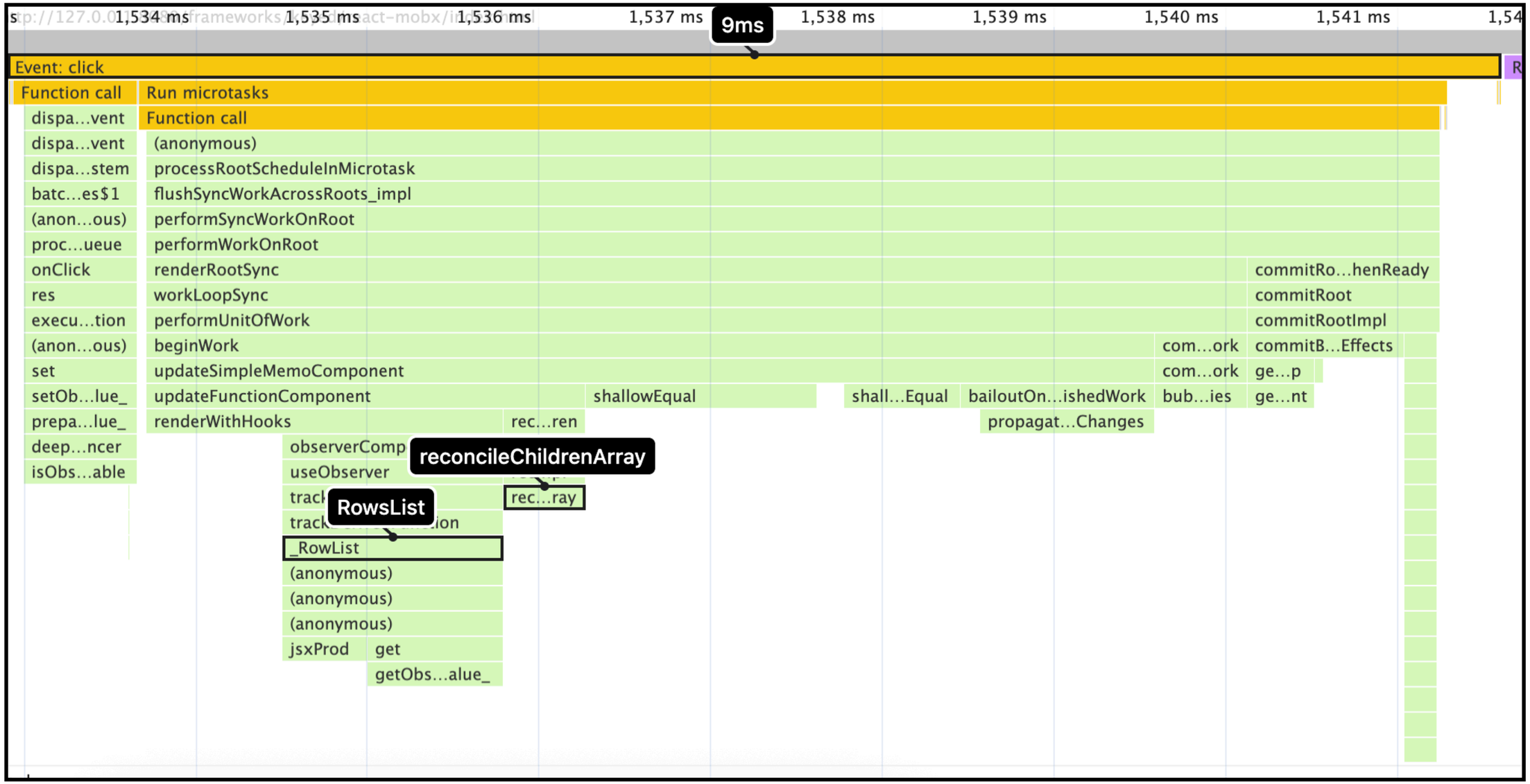
	solid-v1.9.3	svelte-v5.42.1	react-hooks-v19.2.0	react-mobX-v19.0.0 + 6.13.5
select row highlighting a selected row. (5 warmup runs). 4 x CPU slowdown.	14.5 ± 0.7 (1.00)	17.9 ± 1.0 (1.23)	16.6 ± 0.6 (1.14)	20.1 ± 0.8 (1.39) 🤔

Select row (mobX)

```
<table className="table table-hover table-striped test-data">
  <tbody>
    <RowList store={store} />
  </tbody>
</table>
```

```
const _RowList = ({ store }) => {
  return store.rows.map((row) => (
    <Row
      key={row.id}
      data={row}
      onSelect={store.select}
      onDelete={store.delete}
      isSelected={store.selectedRowId === row.id}
    />
  ));
};
```

Select row (mobX)



Select row (mobX) 4 СПОСОБА

https://mobx.js.org/computeds-with-args.html

How can we implement a derivation like `store.isSelected(item.id)`?

```
import * as React from 'react'
import { observer } from 'mobx-react-lite'

const Item = observer(({ item, store }) => (
    <div className={store.isSelected(item.id) ? "selected" : ""}>
        {item.title}
    </div>
))
```

There are four ways in which we can approach this. You can try the solutions below in this CodeSandbox.

Select row (mobX) 4 СПОСОБА

1) store.isSelected(item.id)

2) const isSelected = computed(() => store.isSelected(item.id)).get()

3) store.isSelected -> item.isSelected

4) computedFn in store

Select row (mobX, computed)

2. Close over the arguments

This is a more efficient implementation compared to the original.

```
import * as React from 'react'
import { computed } from 'mobx'
import { observer } from 'mobx-react-lite'

const Item = observer(({ item, store }) => {
    const isSelected = computed(() => store.isSelected(item.id)).get()
    return (
        <div className={isSelected ? "selected" : ""}>
            {item.title}
        </div>
    )
})
```

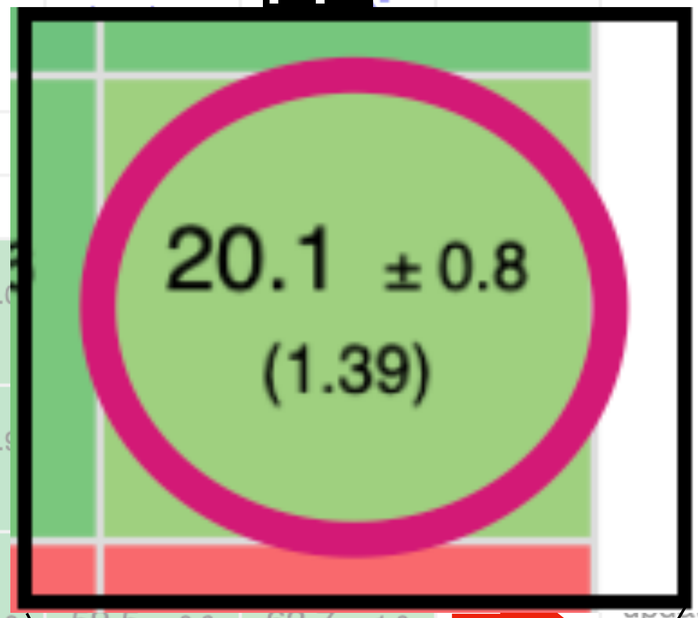
Select row (mobX, computed)

```jsx
const _Row = ({ data, isSelected, onDelete, onSelect }) => {
  return (
    <tr className={isSelected ? 'danger' : ''}>
      <td className="col-md-1">{data.id}</td>
      <td className="col-md-4">
        <a onClick={() => onSelect(data.id)}>{data.label}</a>
      </td>
      <td className="col-md-1">
        <a onClick={() => onDelete(data.id)}>
          <span
            className="glyphicon glyphicon-remove"
            aria-hidden="true"
          ></span>
        </a>
      </td>
      <td className="col-md-6"></td>
    </tr>
  );
};
```

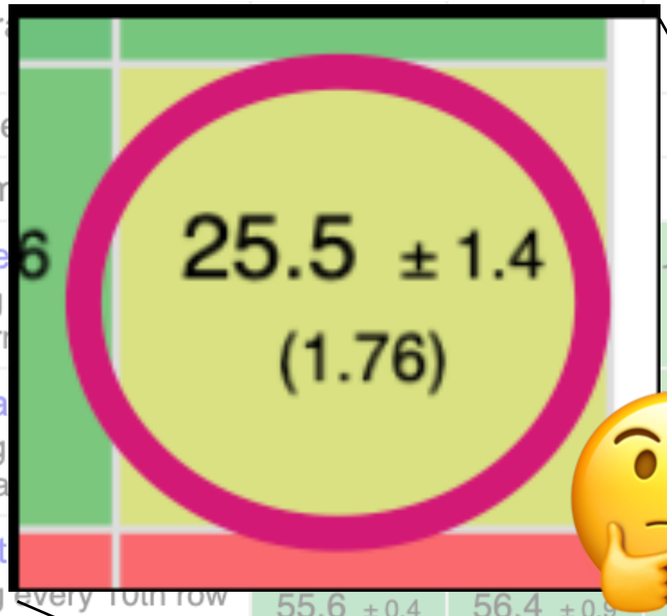
```jsx
const _Row = ({ data, onDelete, onSelect }) => {
  const isSelected = computed(() => store.isSelected(data.id)).get();
  return (
    <tr className={isSelected ? 'danger' : ''}>
      <td className="col-md-1">{data.id}</td>
      <td className="col-md-4">
        <a onClick={() => onSelect(data.id)}>{data.label}</a>
      </td>
      <td className="col-md-1">
        <a onClick={() => onDelete(data.id)}>
          <span
            className="glyphicon glyphicon-remove"
            aria-hidden="true"
          ></span>
        </a>
      </td>
      <td className="col-md-6"></td>
    </tr>
  );
};
```

Select row (mobX, computed)

До



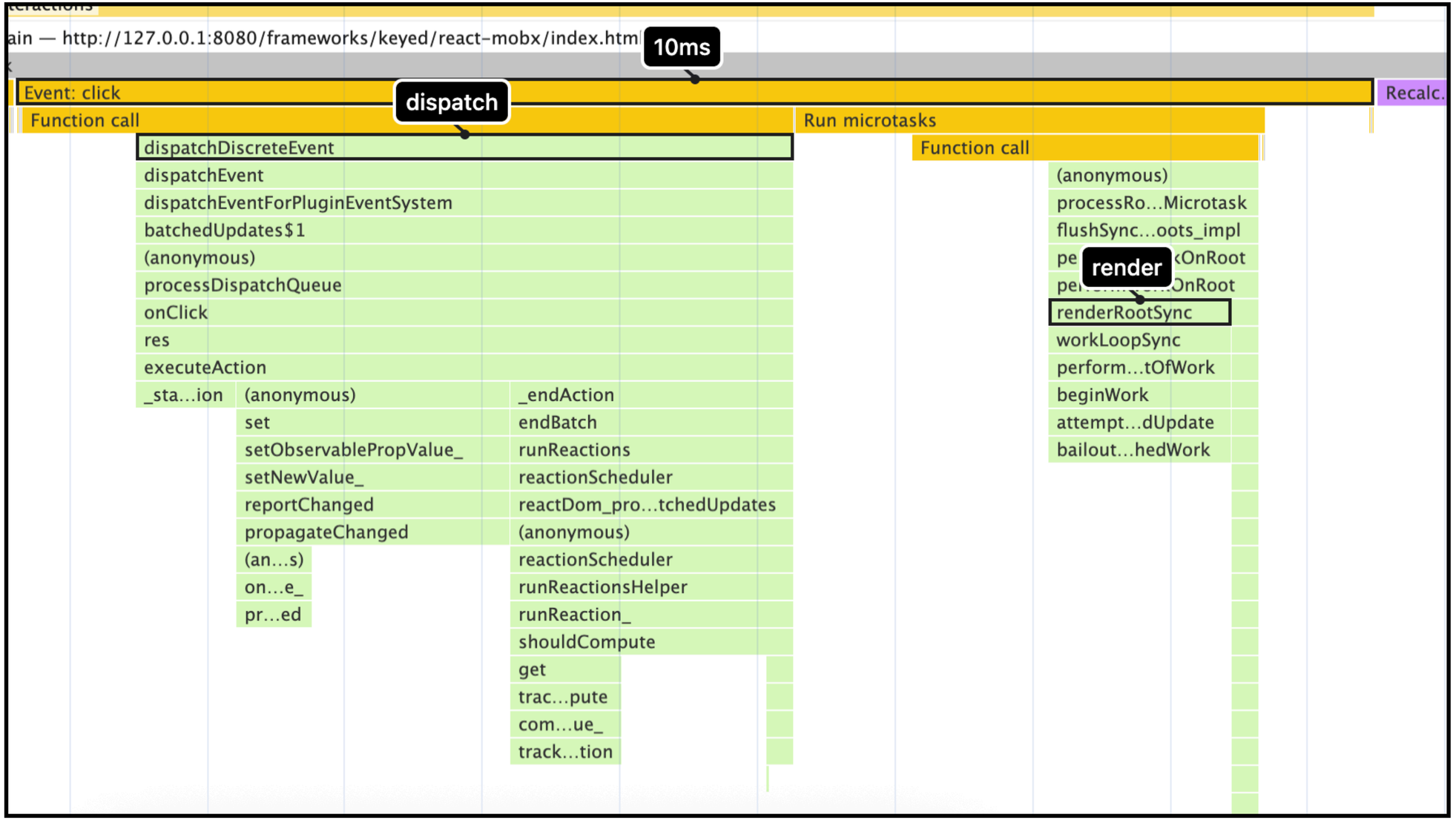
После



Name Duration for...	solid-v1.9.3	svelte-v5.42.1	react-hooks-v19.2.0	react-mobX-v19.0.0 + 6.13.5
Implementation notes				
Implementation link	code	code	code	code
create rows creating 1,000 rows. (5 warmup runs).	90.2 ±1.1 (1.00)	92.8 ±1.0 (1.03)	105.3 ±0.6 (1.17)	114.0 ±1.0 (1.26)
replace all rows updating all 1,000 rows. (5 warmup runs).	97.9 ±1.2 (1.00)	99.4 ±0.9 (1.02)	±0.9	123.9 ±1.2 (1.27)
partial update updating every 10th row for 1,000 row. (3 warmup runs). 4 x CPU slowdown.	55.6 ±0.4 (1.00)	56.4 ±0.9 (1.01)	59.5 ±0.6 (1.07)	62.7 ±1.0 (1.13)
select row highlighting a selected row. (5 warmup runs). 4 x CPU slowdown.	14.5 ±0.7 (1.00)	17.9 ±1.0 (1.23)	16.6 ±0.6 (1.14)	20.1 ±0.8 (1.39)
swap rows swap 2 rows for table with 1,000 rows. (5 warmup runs). 4 x CPU	65.4 ±0.9 (1.01)	64.8 ±1.4 (1.00)	349.7 ±2.7 (5.40)	342.6 ±3.3 (5.29)

Name	svelte-	react-hooks-v19.2.0	react-mobX-v19.0.0 + 6.13.5
for 1,000 row. (3 warmup runs). 4 x CPU slowdown.	55.6 ±0.4 (1.00)	56.4 ±0.9 (1.01)	59.5 ±0.6 (1.07)	62.7 ±1.0 (1.13)
select row highlighting a selected row. (5 warmup runs). 4 x CPU slowdown.	14.5 ±0.7 (1.00)	17.9 ±1.0 (1.23)	16.6 ±0.6 (1.14)	25.5 ±1.4 (1.76)
swap rows swap 2 rows for table with 1,000 rows. (5 warmup runs). 4 x CPU	65.4 ±0.9 (1.01)	64.8 ±1.4 (1.00)	349.7 ±2.7 (5.40)	342.6 ±3.3 (5.29)

Select row (mobX, computed)



Select row (mobX, computedFn)

Name Duration for...	solid-v1.9.3	svelte-v5.42.1	react-hooks-v19.2.0	react-mobX-v19.0.0 + 6.13.5
Implementation notes				
Implementation link	code	code	code	code
create rows creating 1,000 rows. (5 warmup runs).	90.2 ±1.1 (1.00)	92.8 ± (1.03		
replace all rows updating all 1,000 rows. (5 warmup runs).	97.9 ±1.2 (1.00)	99.4 ± (1.02		
partial update updating every 10th row for 1,000 row. (3 warmup runs). 4 x CPU slowdown.	55.6 ±0.4 (1.00)	56.4 ±0.9 (1.01)	59.5 ±0.6 (1.07)	62.7 ±1.0 (1.13)
select row highlighting a selected row. (5 warmup runs). 4 x CPU slowdown.	14.5 ±0.7 (1.00)	17.9 ±1.0 (1.23)	16.6 ±0.6 (1.14)	25.5 ±1.4 (1.76)
swap rows swap 2 rows for table with 1,000 rows. (5 warmup runs). 4 x CPU slowdown.	65.4 ±0.9 (1.01)	64.8 ±1.4 (1.00)	349.7 ±2.7 (5.40)	342.6 ±3.3 (5.29)

Computed

25.5 ±1.4
(1.76)

Name Duration f	solid- 3	svelte-v5.42.1	react-hooks-v19.2.0	react-mobX-v19.0.0 + 6.13.5
Impl				
Imp		code	code	code
crea 6		92.8 ±1.0 (1.03)	105.3 ±0.6 (1.17)	114.0 ±1.0 (1.26)
re upda (5		99.4 ±0.9 (1.02)	118.8 ±0.9 (1.21)	123.9 ±1.2 (1.27)
partial update updating every 10th row for 1,000 row. (3 warmup runs). 4 x CPU slowdown.	55.6 ±0.4 (1.00)	56.4 ±0.9 (1.01)	59.5 ±0.6 (1.07)	62.7 ±1.0 (1.13)
select row highlighting a selected row. (5 warmup runs). 4 x CPU slowdown.	14.5 ±0.7 (1.00)	17.9 ±1.0 (1.23)	16.6 ±0.6 (1.14)	23.5 ±0.7 (1.62)
swap rows swap 2 rows for table with 1,000 rows. (5 warmup runs). 4 x CPU slowdown.	65.4 ±0.9 (1.01)	64.8 ±1.4 (1.00)	349.7 ±2.7 (5.40)	342.6 ±3.3 (5.29)

ComputedFn

23.5 ±0.7
(1.62)

Select row (mobX, not computed)

1. Derivations don't *need* to be `computed`

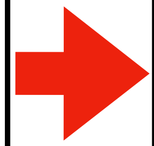
A function doesn't need to be marked as `computed` in order for MobX to track it. The above example would already work completely fine out of the box. It is important to realize that computed values are only *caching points*. If the derivations are pure (and they should be), having a getter or function without `computed` doesn't change the behavior, it is just slightly less efficient.

The above example works fine despite `isSelected` not being a `computed`. The `observer` component will detect and subscribe to any observables that were read by `isSelected` because the function executes as part of rendering that is tracked.

It is good to realize that all `Item` components, in this case, will respond to future selection changes, as they all subscribe directly to the observables that capture the selection. This is a worst-case example. In general, it is completely fine to have unmarked functions that derive information, and this is a good default strategy until numbers prove anything else should be done.

Select row (mobX, not computed)

```
const _Row = ({ data, isSelected, onDelete, onSelect }) => {
  return (
    <tr className={isSelected ? 'danger' : ''}>
      <td className="col-md-1">{data.id}</td>
      <td className="col-md-4">
        <a onClick={() => onSelect(data.id)}>{data.label}</a>
      </td>
      <td className="col-md-1">
        <a onClick={() => onDelete(data.id)}>
          <span
            className="glyphicon glyphicon-remove"
            aria-hidden="true"
          ></span>
        </a>
      </td>
      <td className="col-md-6"></td>
    </tr>
  );
};
```

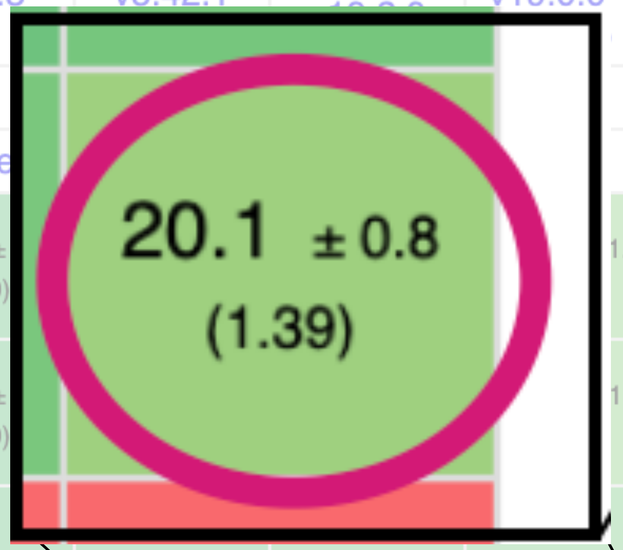


```
const _Row = ({ data, onDelete, onSelect }) => {
  const isSelected = store.isSelected(data.id);
  return (
    <tr className={isSelected ? 'danger' : ''}>
      <td className="col-md-1">{data.id}</td>
      <td className="col-md-4">
        <a onClick={() => onSelect(data.id)}>{data.label}</a>
      </td>
      <td className="col-md-1">
        <a onClick={() => onDelete(data.id)}>
          <span
            className="glyphicon glyphicon-remove"
            aria-hidden="true"
          ></span>
        </a>
      </td>
      <td className="col-md-6"></td>
    </tr>
  );
};
```

Select row (mobX, not computed)

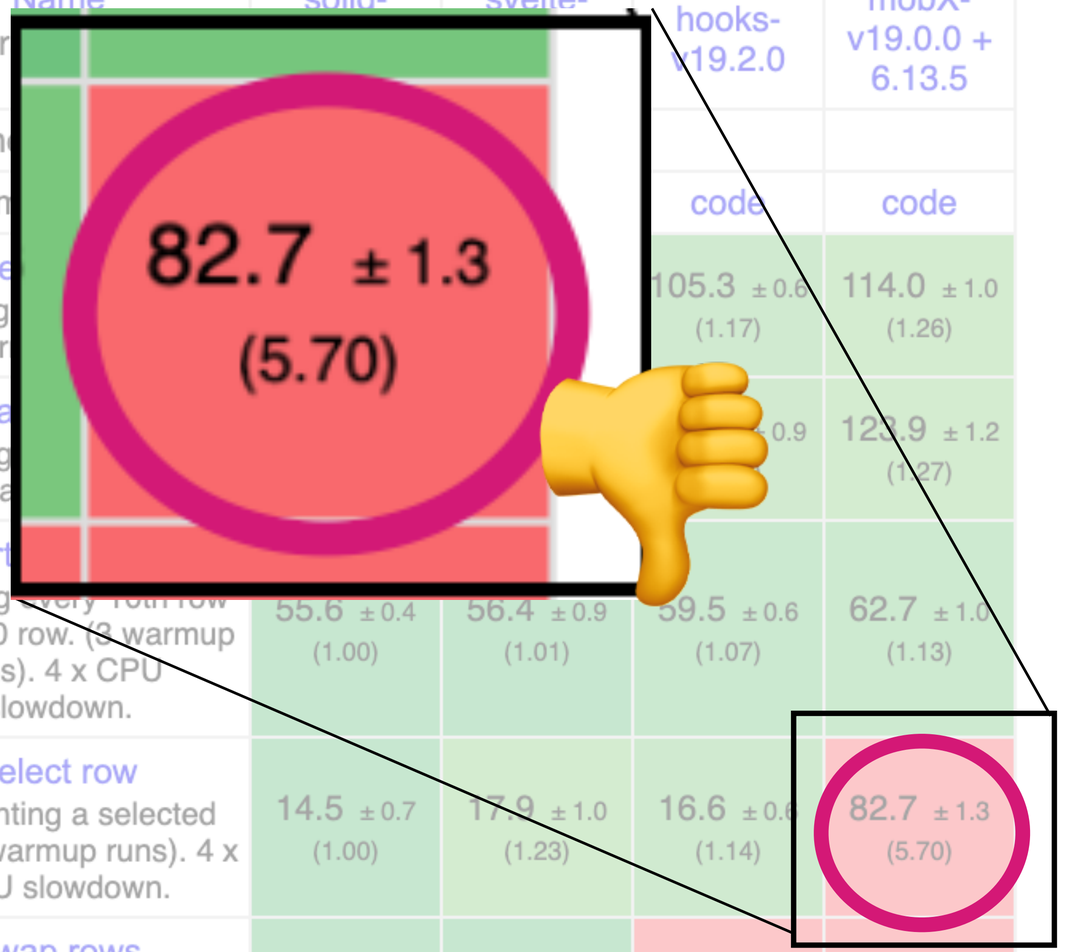
До

Name Duration for...	solid-v1.9.3	svelte-v5.42.1	r… hooks-…	react-mobX-v19.0.0 + …
Implementation notes				
Implementation link	code			
create rows creating 1,000 rows. (5 warmup runs).	90.2 ± … (1.00)			1.0
replace all rows updating all 1,000 rows. (5 warmup runs).	97.9 ± … (1.00)			1.2
partial update updating every 10th row for 1,000 row. (3 warmup runs). 4 x CPU slowdown.	55.6 ± 0.4 (1.00)	56.4 ± 0.9 (1.01)	59.5 ± 0.6 (1.07)	62.7 ± 1.0 (1.13)
select row highlighting a selected row. (5 warmup runs). 4 x CPU slowdown.	14.5 ± 0.7 (1.00)	17.9 ± 1.0 (1.23)	16.6 ± 0.6 (1.14)	20.1 ± 0.8 (1.39)
swap rows swap 2 rows for table with 1,000 rows. (5 warmup runs). 4 x CPU	65.4 ± 0.9 (1.01)	64.8 ± 1.4 (1.00)	349.7 ± 2.7 (5.40)	342.6 ± 3.3 (5.29)



После

Name Dur…	solid-…	svelte-…	react-hooks-v19.2.0	react-mobX-v19.0.0 + 6.13.5
Implem…				
Implen…			code	code
cre… creating … war…			105.3 ± 0.6 (1.17)	114.0 ± 1.0 (1.26)
repla… updating … (5 wa…			…0.9	123.9 ± 1.2 (1.27)
part… updating … for 1,000 row. (3 warmup runs). 4 x CPU slowdown.	55.6 ± 0.4 (1.00)	56.4 ± 0.9 (1.01)	59.5 ± 0.6 (1.07)	62.7 ± 1.0 (1.13)
select row highlighting a selected row. (5 warmup runs). 4 x CPU slowdown.	14.5 ± 0.7 (1.00)	17.9 ± 1.0 (1.23)	16.6 ± 0.6 (1.14)	82.7 ± 1.3 (5.70)
swap rows swap 2 rows for table with 1,000 rows. (5 warmup runs). 4 x CPU	65.4 ± 0.9 (1.01)	64.8 ± 1.4 (1.00)	349.7 ± 2.7 (5.40)	342.6 ± 3.3 (5.29)



Select row (mobX, not computed)

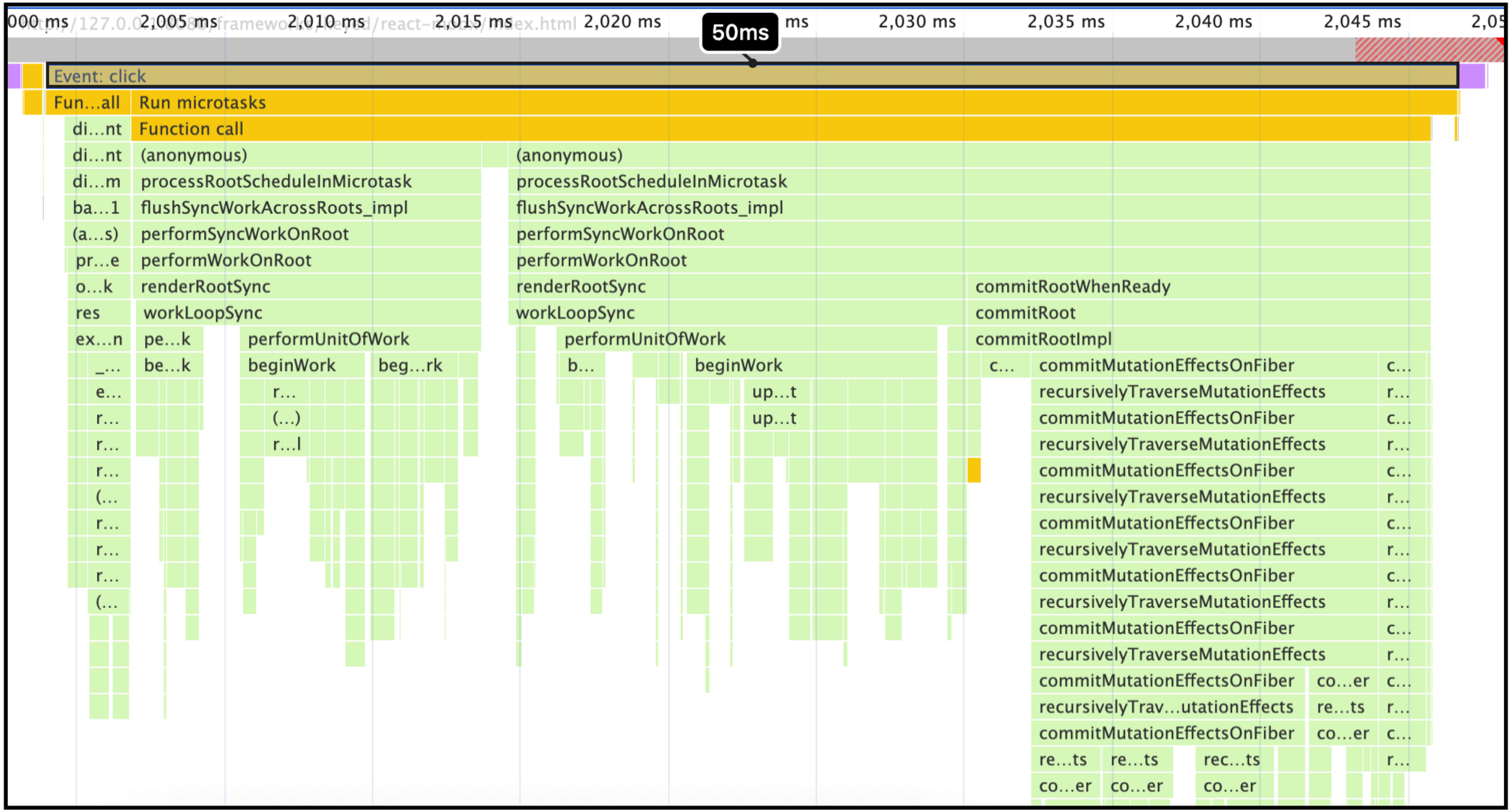
1. Derivations don't *need* to be `computed`

A function doesn't need to be marked as `computed` in order for MobX to track it. The above example would already work completely fine out of the box. It is important to realize that computed values are only *caching points*. If the derivations are pure (and they should be), having a getter or function without `computed` doesn't change the behavior, it is just slightly less efficient.

The above example works fine despite `isSelected` not being a `computed`. The `observer` component will detect and subscribe to any observables that were read by `isSelected` because the function executes as part of rendering that is tracked.

It is good to realize that all `Item` components, in this case, will respond to future selection changes, as they all subscribe directly to the observables that capture the selection. This is a worst-case example. In general, it is completely fine to have unmarked functions that derive information, and this is a good default strategy until numbers prove anything else should be done.

Select row (mobX, not computed)



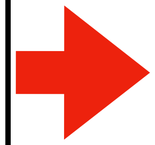
Select row (mobX, move state)

3. Move the state

In this specific case the selection could also be stored as an `isSelected` observable on the `Item`. The selection in the store could then be expressed as a `computed` rather than an observable: `get selection() { return this.items.filter(item => item.isSelected) }`, and we don't need `isSelected` anymore.

Select row (mobX, move state)

```jsx
const _Row = ({ data, isSelected, onDelete, onSelect }) => {
  return (
    <tr className={isSelected ? 'danger' : ''}>
      <td className="col-md-1">{data.id}</td>
      <td className="col-md-4">
        <a onClick={() => onSelect(data.id)}>{data.label}</a>
      </td>
      <td className="col-md-1">
        <a onClick={() => onDelete(data.id)}>
          <span
            className="glyphicon glyphicon-remove"
            aria-hidden="true"
          ></span>
        </a>
      </td>
      <td className="col-md-6"></td>
    </tr>
  );
};
```



```jsx
const _Row = ({ data, onDelete, onSelect }) => {
  return (
    <tr className={data.isSelected ? 'danger' : ''}>
      <td className="col-md-1">{data.id}</td>
      <td className="col-md-4">
        <a onClick={() => onSelect(data.id)}>{data.label}</a>
      </td>
      <td className="col-md-1">
        <a onClick={() => onDelete(data.id)}>
          <span
            className="glyphicon glyphicon-remove"
            aria-hidden="true"
          ></span>
        </a>
      </td>
      <td className="col-md-6"></td>
    </tr>
  );
};
```

Move state

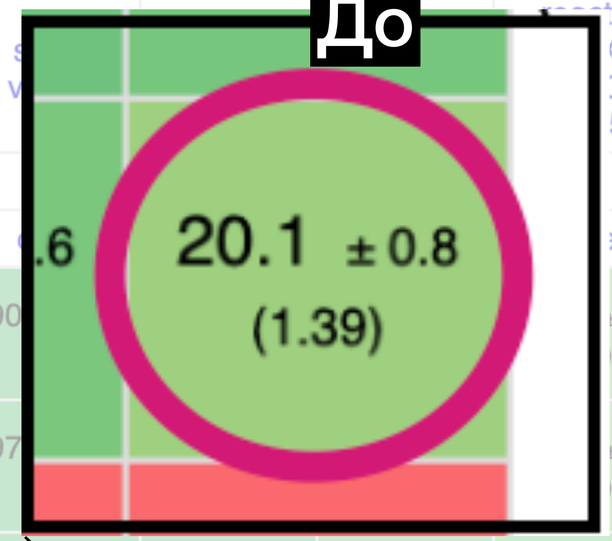
Select row (mobX, move state)

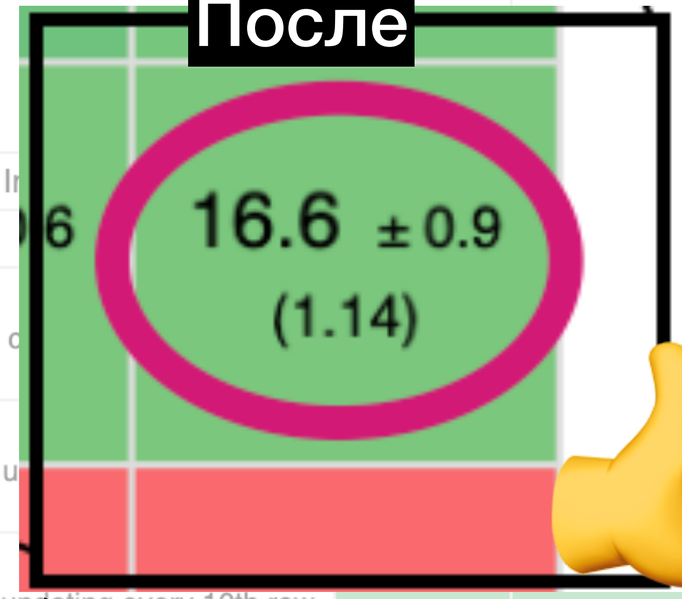
До

Name Duration for...				
Implementation notes				
Implementation link				
create rows creating 1,000 rows. (5 warmup runs).	90			± 1.0
replace all rows updating all 1,000 rows. (5 warmup runs).	97			± 1.2
partial update updating every 10th row for 1,000 row. (3 warmup runs). 4 x CPU slowdown.	55.6 ± 0.4 (1.00)	56.4 ± 0.9 (1.01)	59.5 ± 0.6 (1.07)	62.7 ± 1.0 (1.13)
select row highlighting a selected row. (5 warmup runs). 4 x CPU slowdown.	14.5 ± 0.7 (1.00)	17.9 ± 1.0 (1.23)	16.6 ± 0.6 (1.14)	20.1 ± 0.8 (1.39)
swap rows swap 2 rows for table with 1,000 rows. (5 warmup runs). 4 x CPU slowdown.	65.4 ± 0.9 (1.01)	64.8 ± 1.4 (1.00)	349.7 ± 2.7 (5.40)	342.6 ± 3.3 (5.29)
remove row	47.9 ± 1.1	47.3 ± 1.1	47.3 ± 1.4	49.8 ± 0.9

После

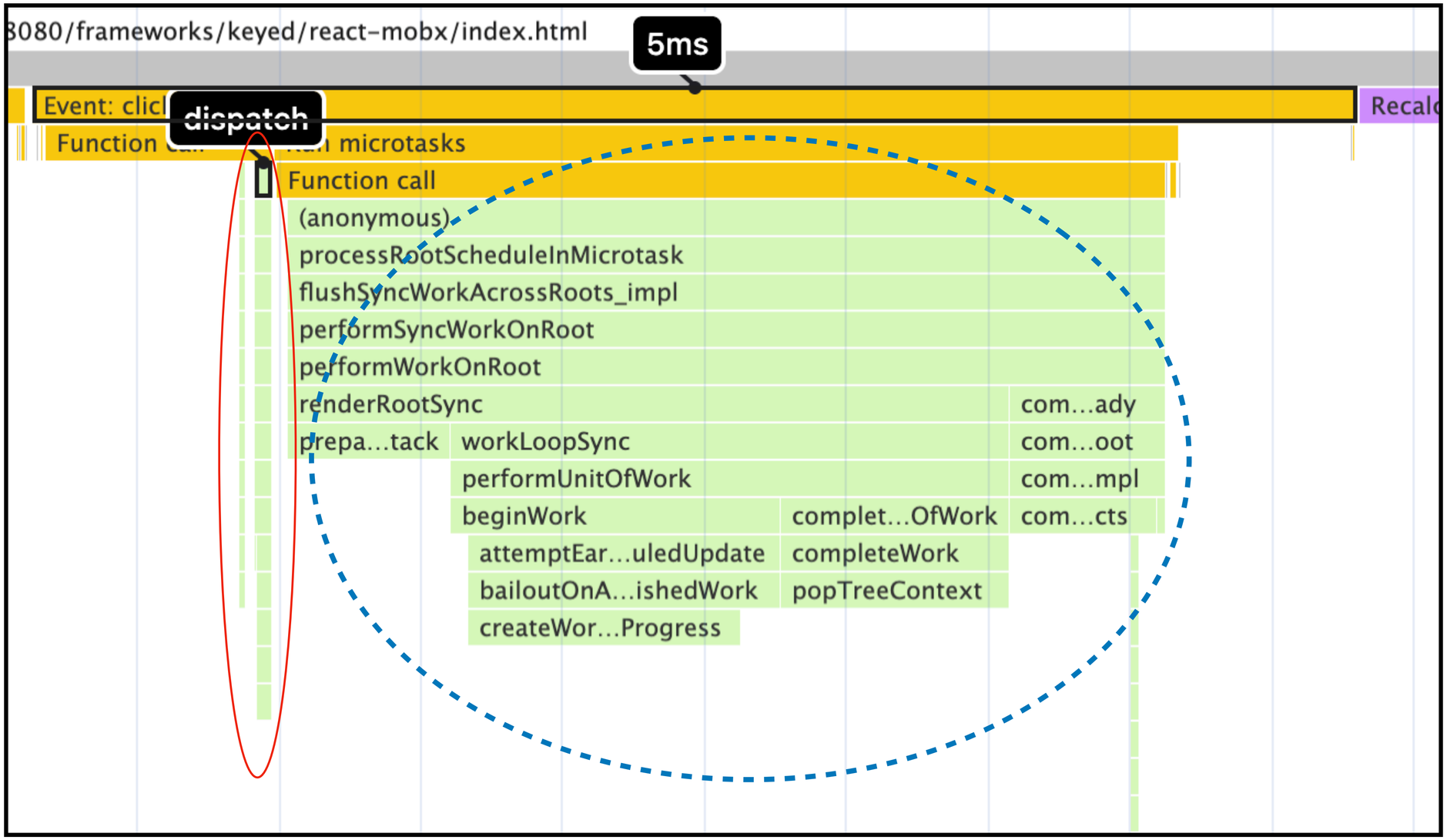
Name Duration for...	react-hooks-v19.2.0	react-mobX-v19.0.0 + 6.13.5
Implementation notes		
Implementation link	code	code
create rows creating 1,000 rows. (5 warmup runs).	105.3 ± 0.6 (1.17)	114.0 ± 1.0 (1.26)
replace all rows updating all 1,000 rows. (5 warmup runs).	± 0.9	123.9 ± 1.2 (1.37)
partial update updating every 10th row for 1,000 row. (3 warmup runs). 4 x CPU slowdown.	59.5 ± 0.6 (1.07)	62.7 ± 1.0
select row highlighting a selected row. (5 warmup runs). 4 x CPU slowdown.	16.6 ± 0.6 (1.14)	16.6 ± 0.9 (1.14)
swap rows swap 2 rows for table with 1,000 rows. (5 warmup runs). 4 x CPU slowdown.	349.7 ± 2.7 (5.40)	342.6 ± 3.3 (5.29)
remove row	47.3 ± 1.4	49.8 ± 0.9





Move state

Select row (mobX, move state)

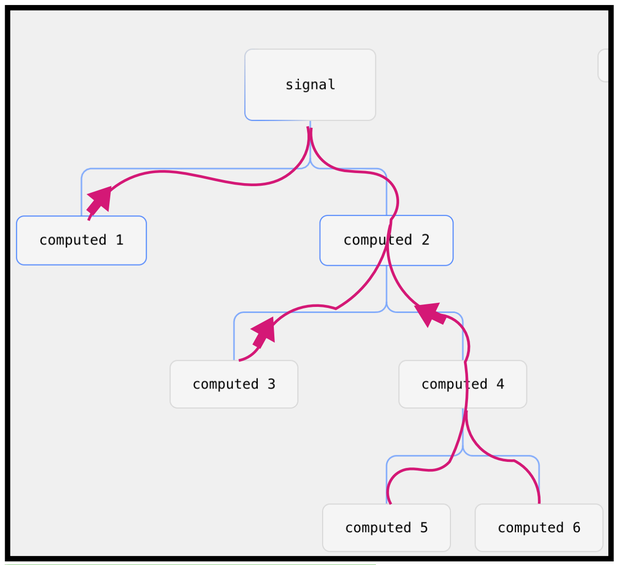


Select row

- Implementations should keep the selected rows in the state (i.e. not a flag for each row, but one reference, id or index for the table) and use that information for rendering. Keeping a selection flag for each row might be faster, but it's considered bad style. Thus those implementations get note #800.

Итого

Svelte

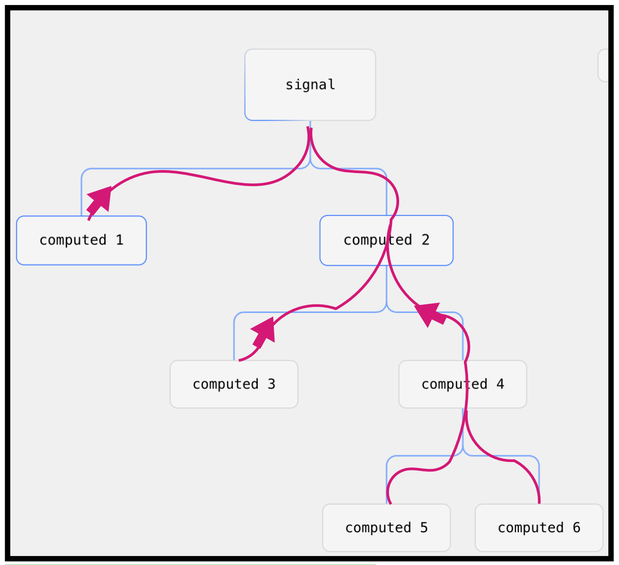


17.9 ± 1.0

(1.23)

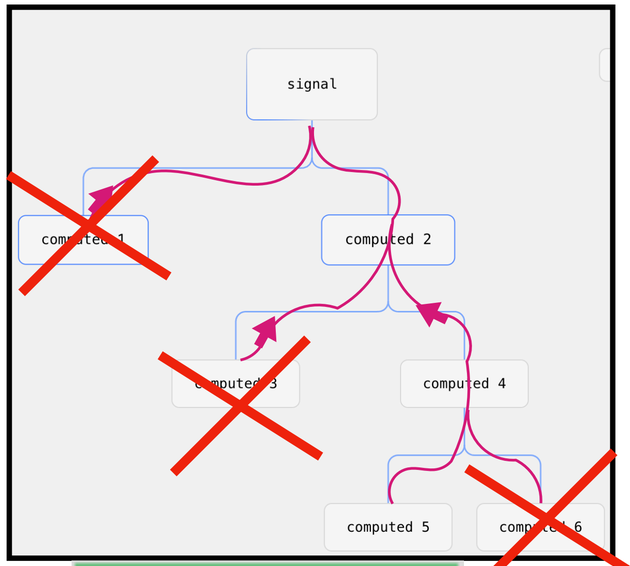
Итого

Svelte



17.9 ± 1.0
(1.23)

Solid

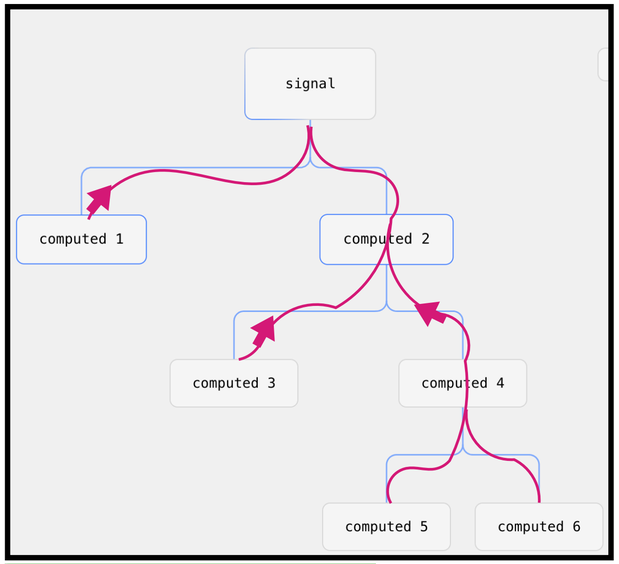


14.5 ± 0.7
(1.00)

17.7 ± 0.7
(1.13)

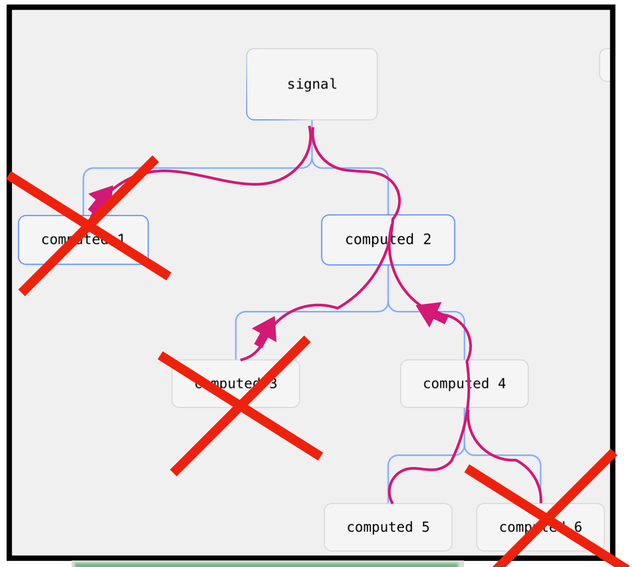
Итого

Svelte



17.9 ± 1.0
(1.23)

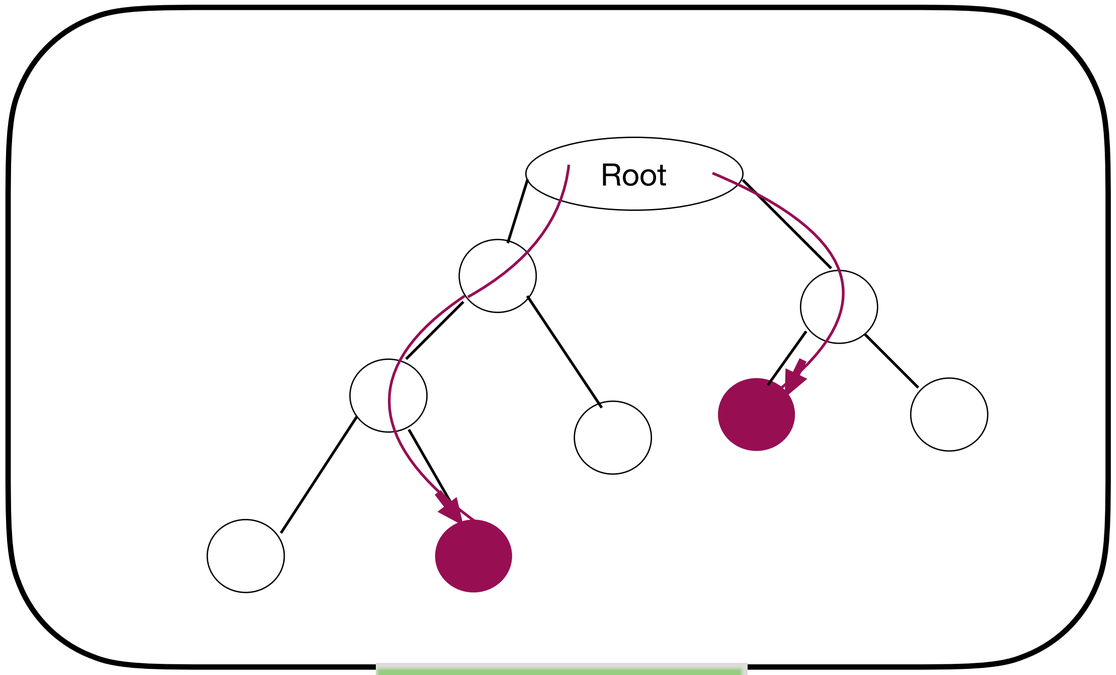
Solid



14.5 ± 0.7
(1.00)

17.7 ± 0.7
(1.13)

React

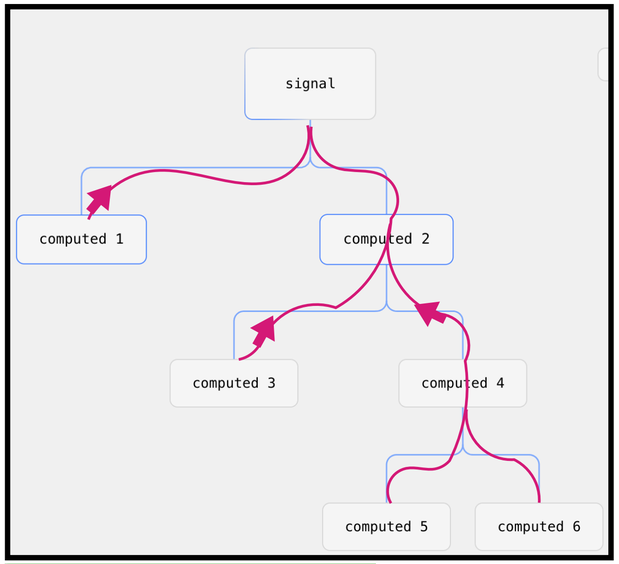


19.2 ± 0.8
(1.

18.1 ± 0.6

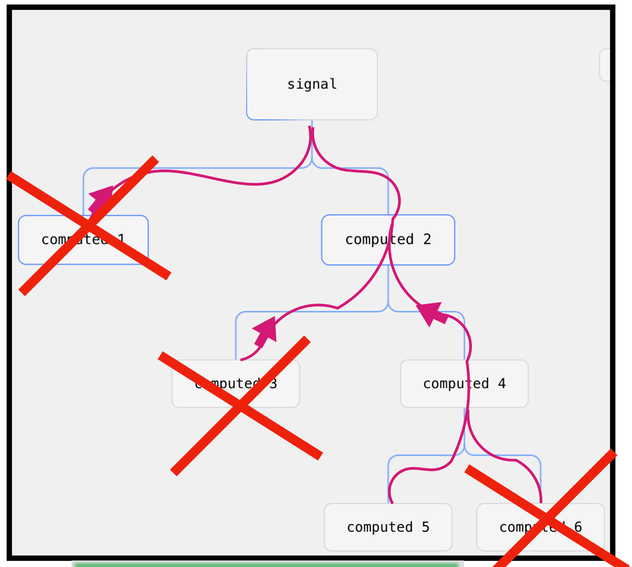
Итого

Svelte



17.9 ± 1.0
(1.23)

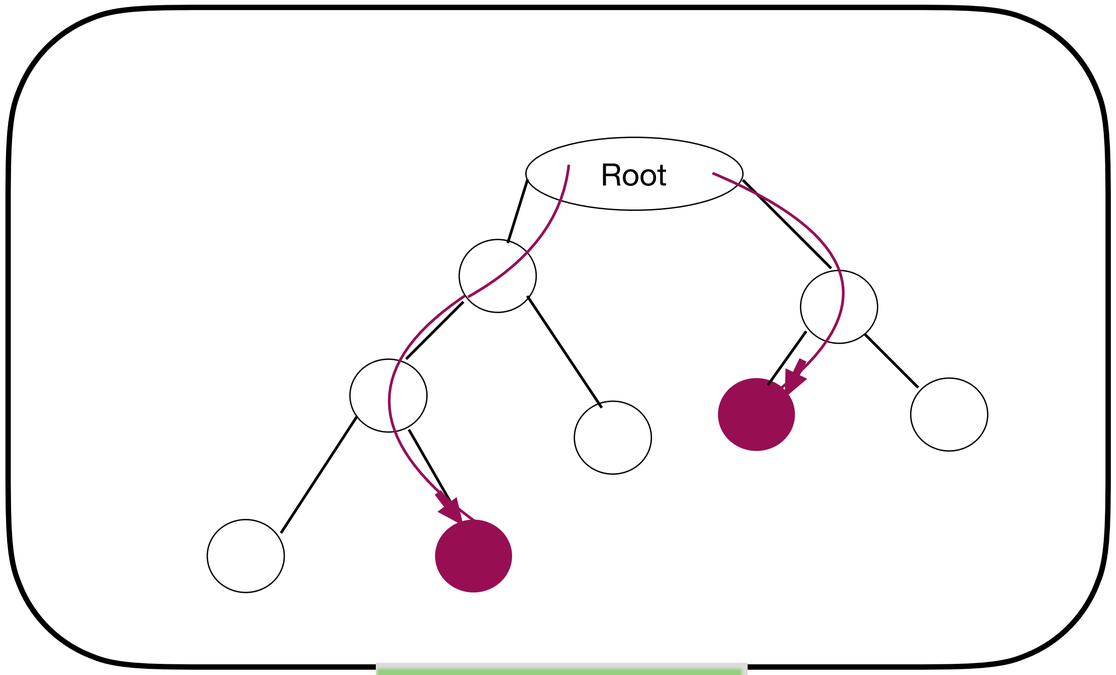
Solid



14.5 ± 0.7
(1.00)

17.7 ± 0.7
(1.13)

React



19.2 ± 0.8
(1.

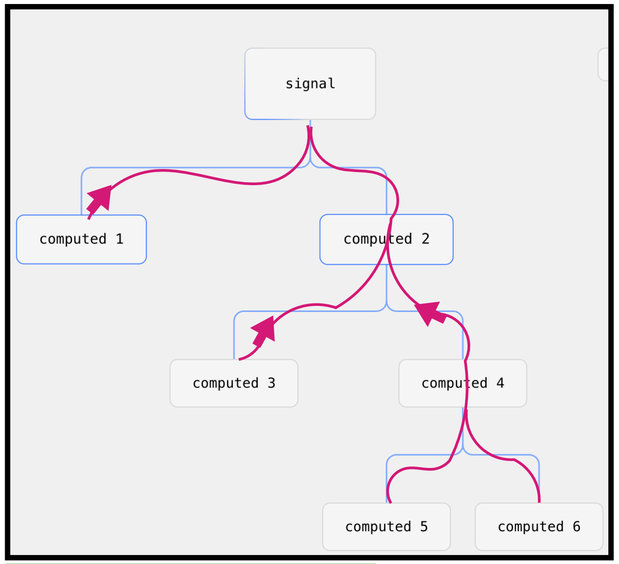
18.1 ± 0.6

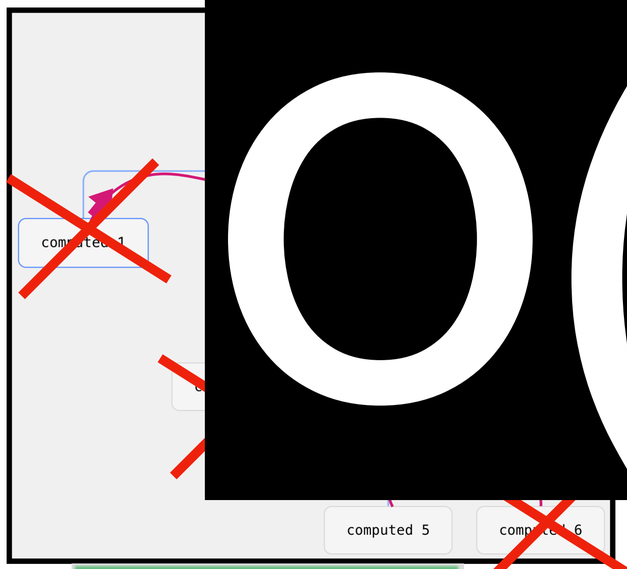
Итого

Svelte

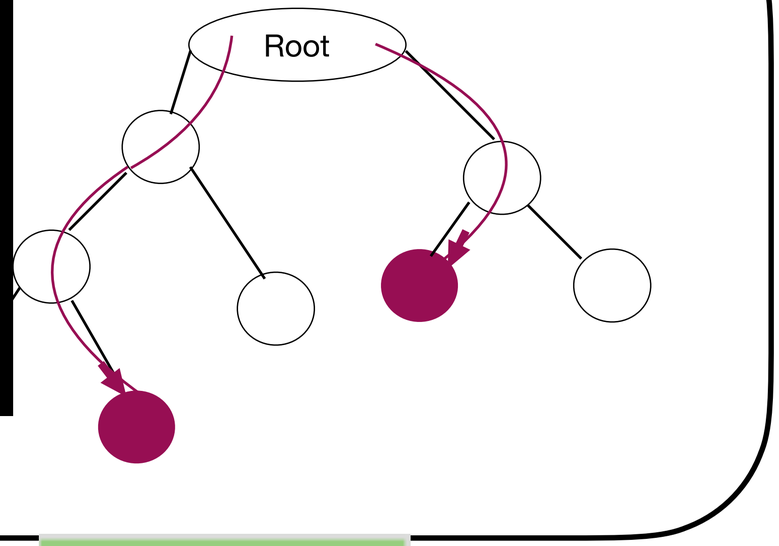
Solid

React





O(n)



17.9 ± 1.0
(1.23)

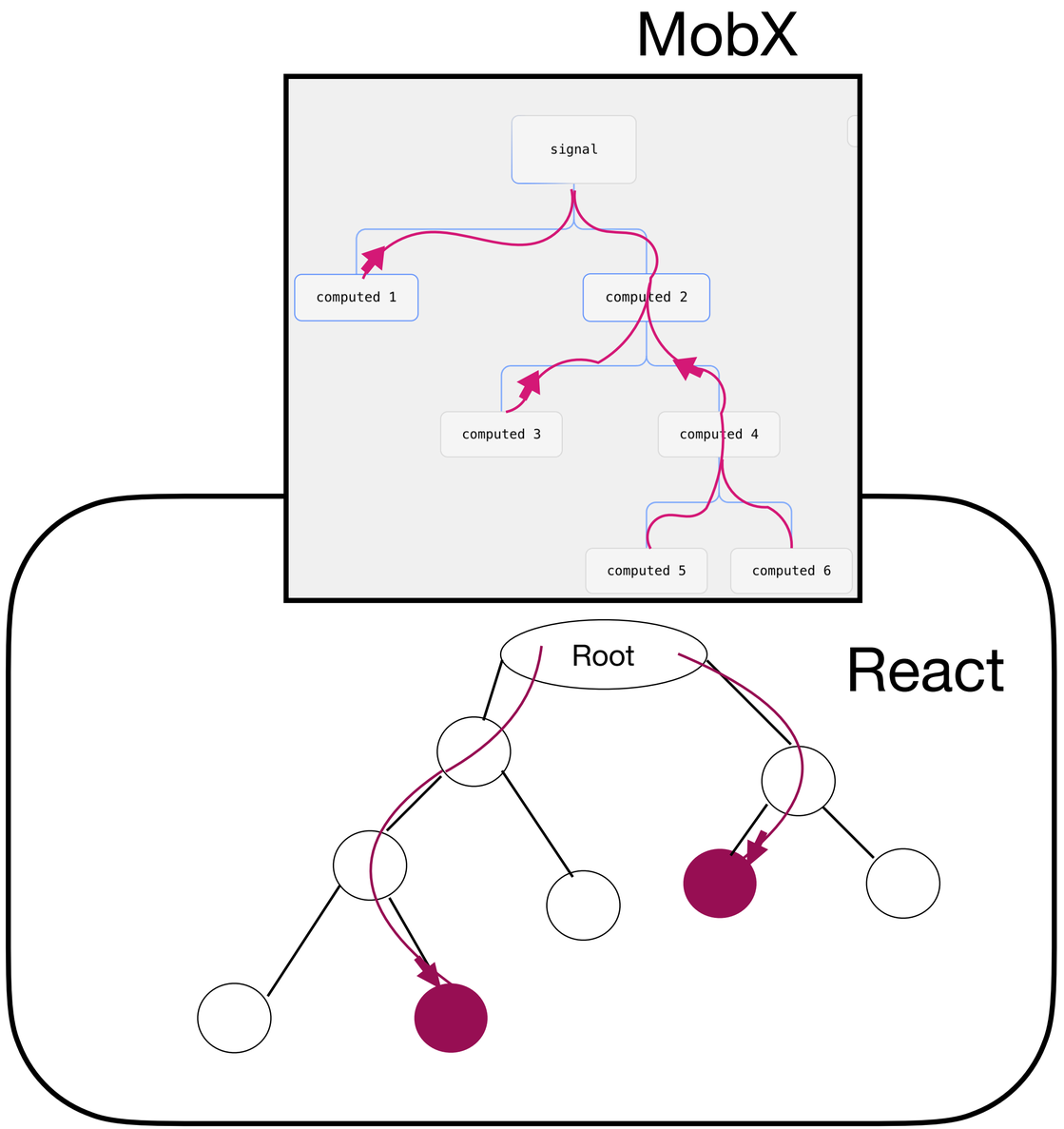
14.5 ± 0.7
(1.00)

17.7 ± 0.7
(1.13)

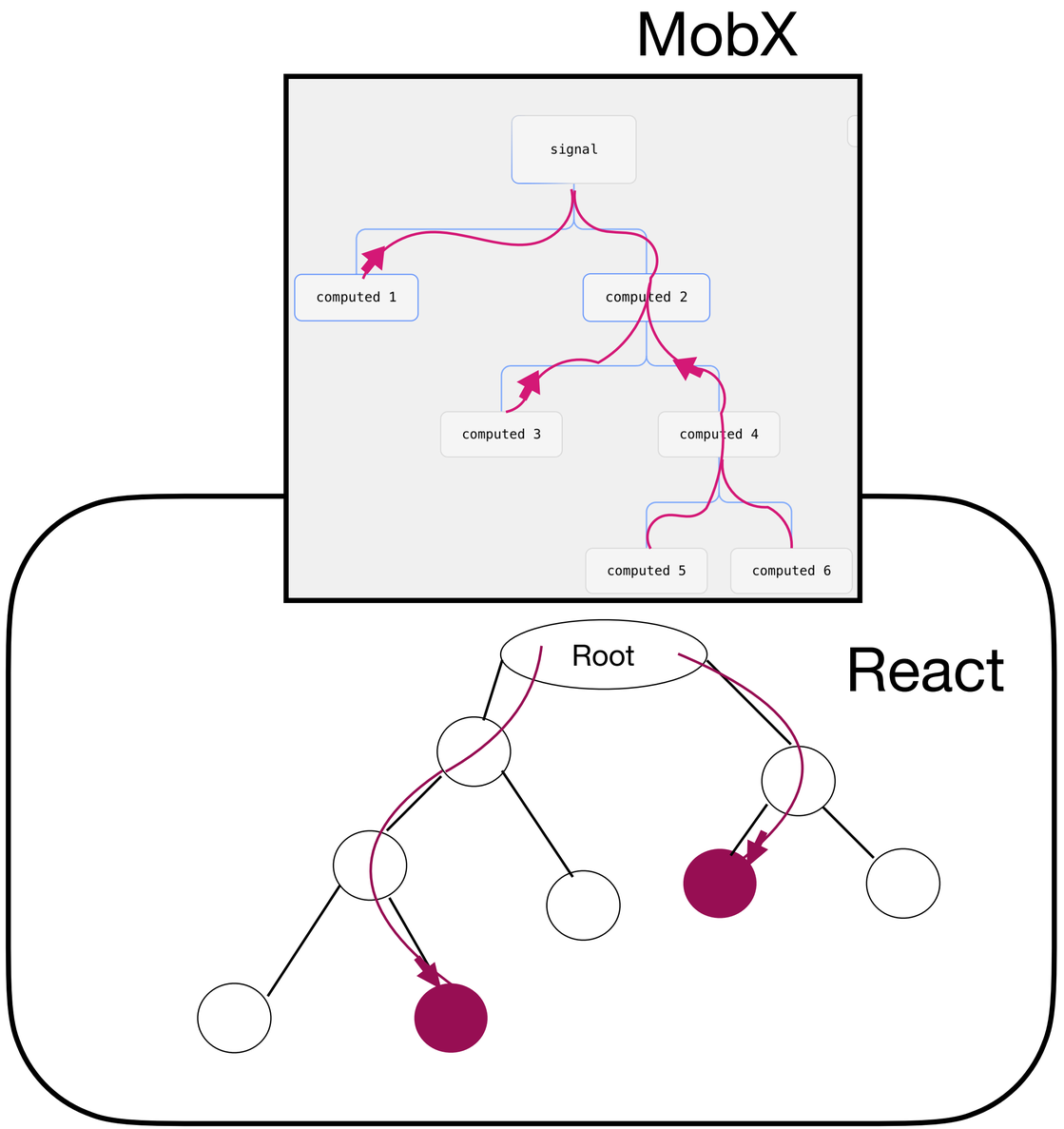
19.2 ± 0.8

18.1 ± 0.6

Итого

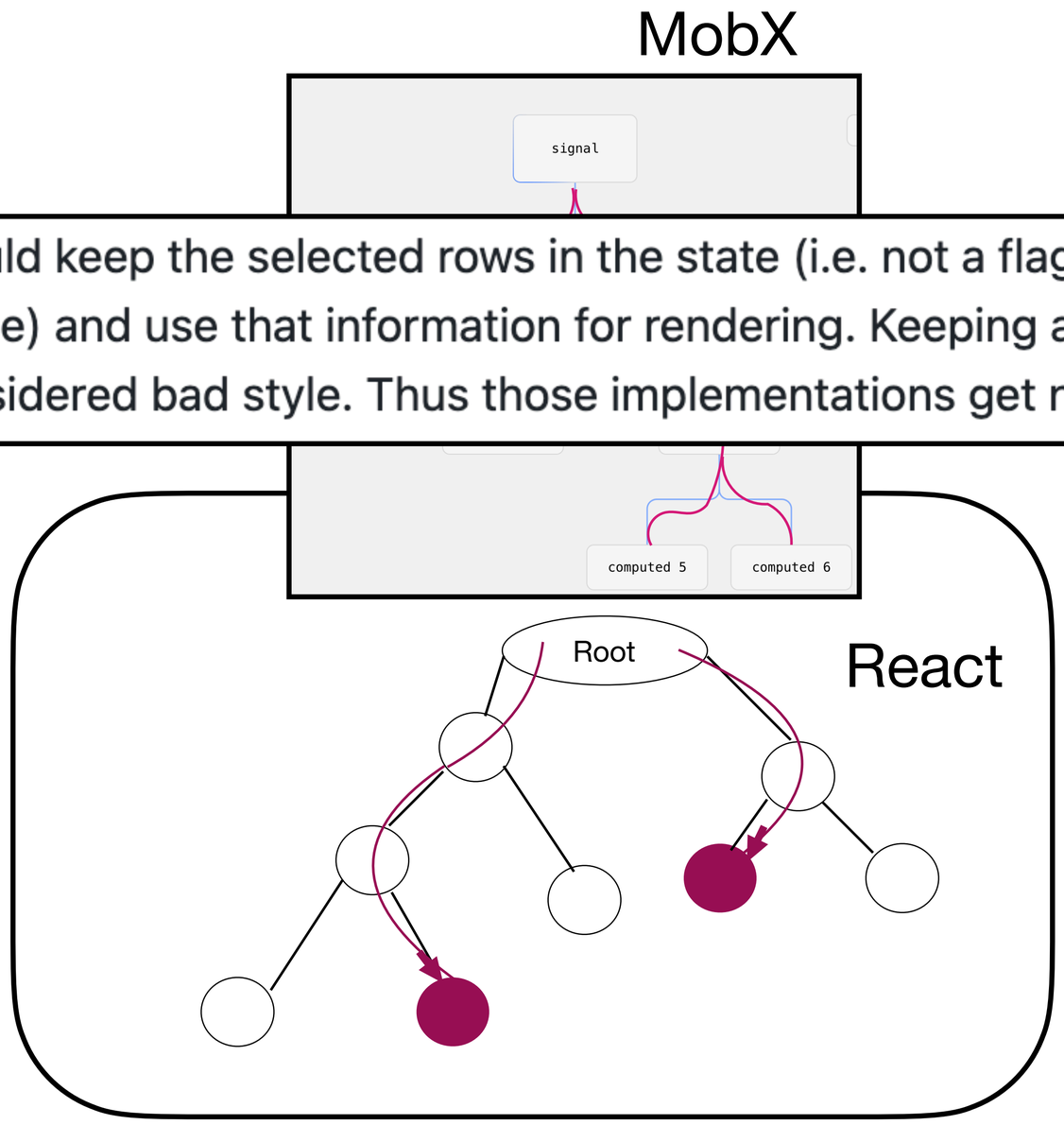

MobX
signal
computed 1
computed 2
computed 3
computed 4
computed 5
computed 6
Root
React

Итого


MobX
signal
computed 1
computed 2
computed 3
computed 4
computed 5
computed 6
Root
React

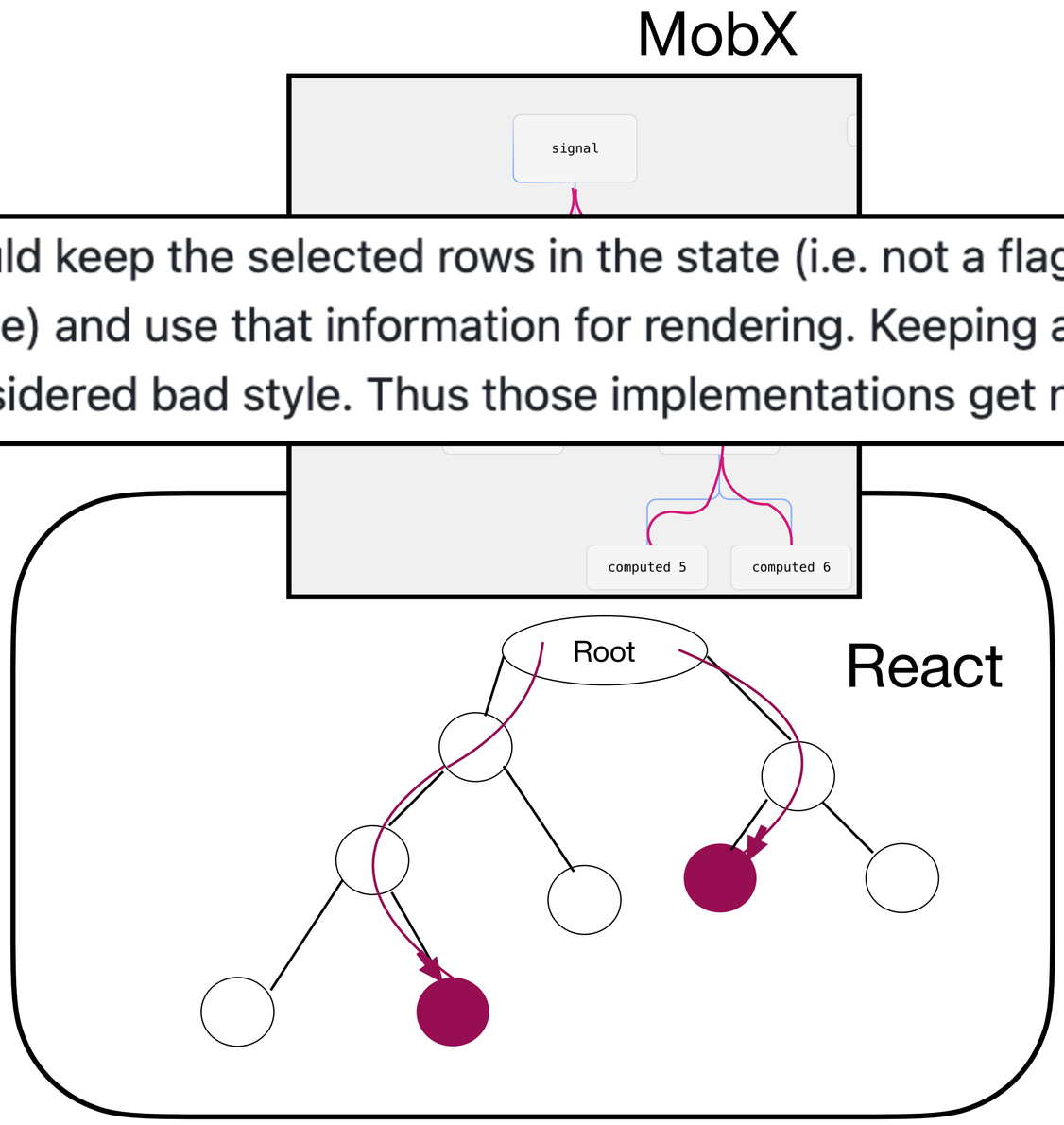
Итого

- Implementations should keep the selected rows in the state (i.e. not a flag for each row, but one reference, id or index for the table) and use that information for rendering. Keeping a selection flag for each row might be faster, but it's considered bad style. Thus those implementations get note #800.

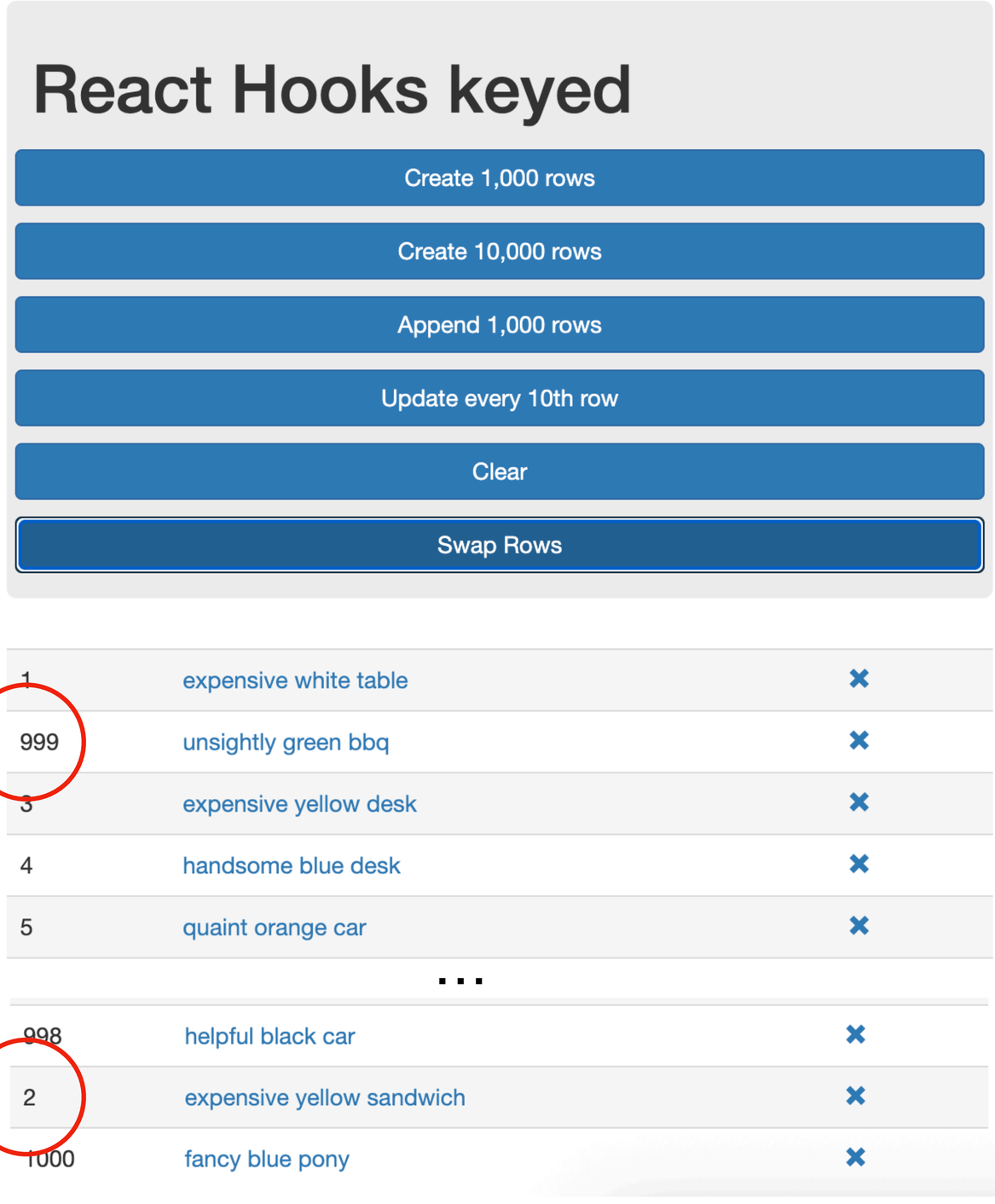


Итого

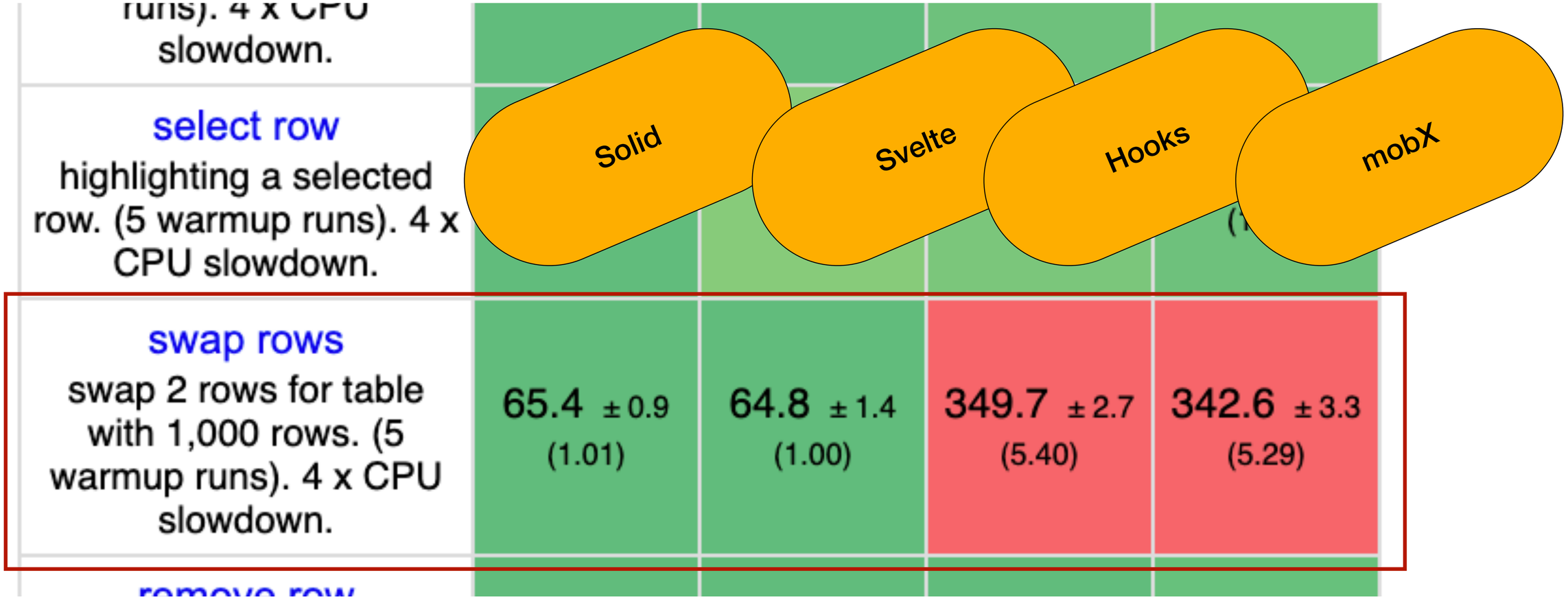
- Implementations should keep the selected rows in the state (i.e. not a flag for each row, but one reference, id or index for the table) and use that information for rendering. Keeping a selection flag for each row might be faster, but it's considered bad style. Thus those implementations get note #800.



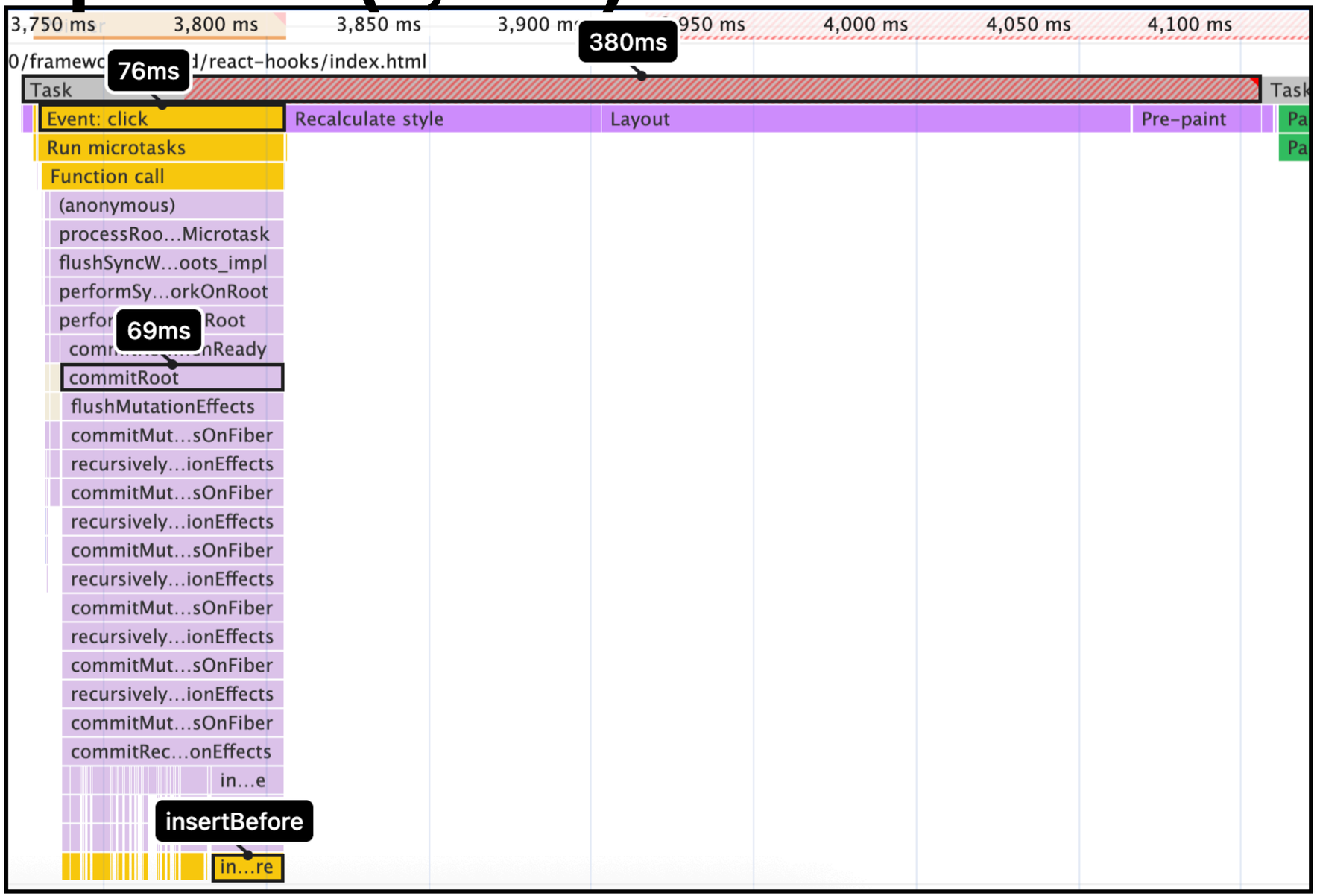
Swap rows


React Hooks keyed
Create 1,000 rows
Create 10,000 rows
Append 1,000 rows
Update every 10th row
Clear
Swap Rows
1 expensive white table
999 unsightly green bbq
3 expensive yellow desk
4 handsome blue desk
5 quaint orange car
998 helpful black car
2 expensive yellow sandwich
1000 fancy blue pony

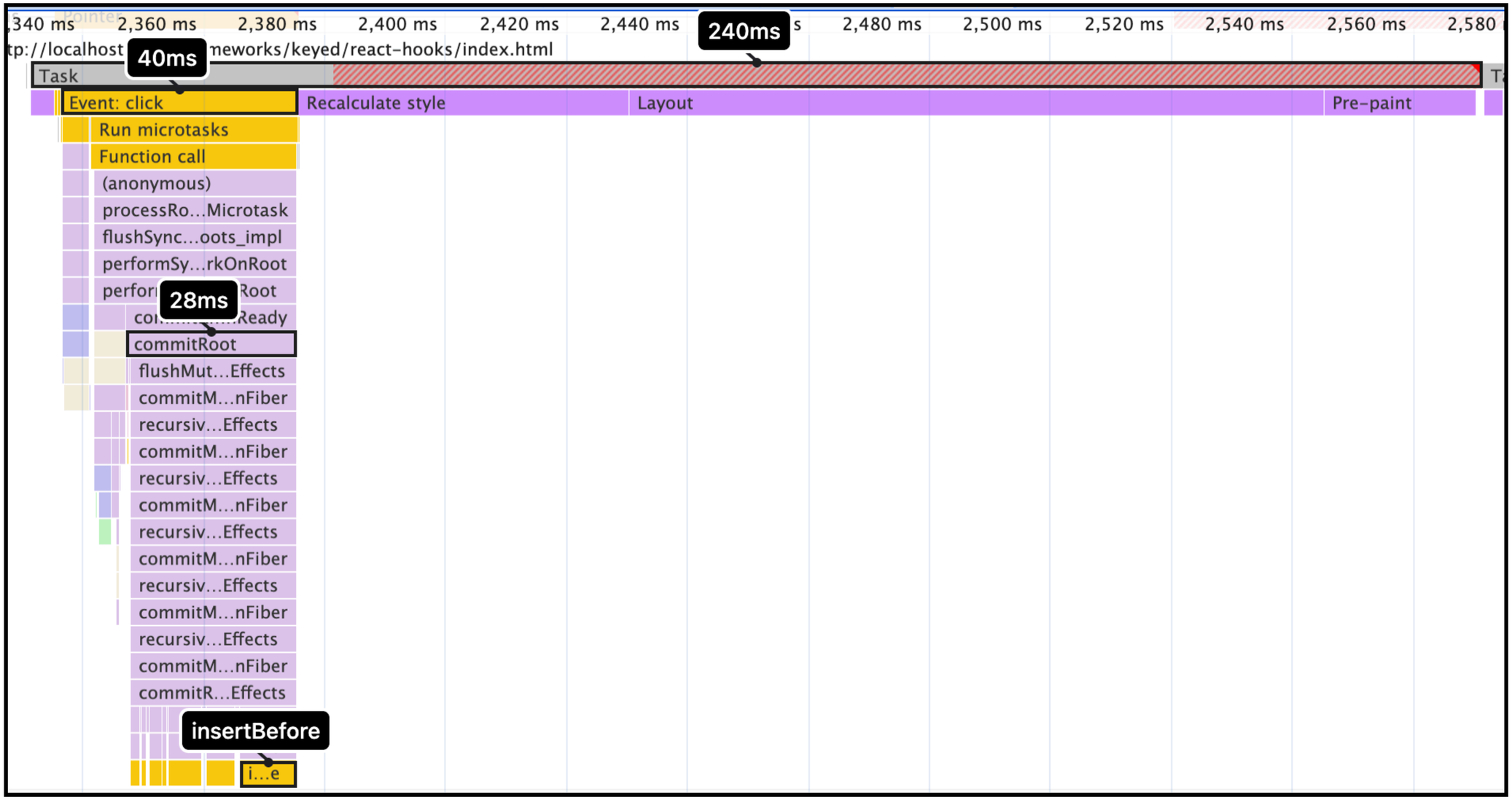
Swap rows


runs). 4 x CPU slowdown.
select row
highlighting a selected row. (5 warmup runs). 4 x CPU slowdown.
swap rows
swap 2 rows for table with 1,000 rows. (5 warmup runs). 4 x CPU slowdown.
65.4 ± 0.9 (1.01)
64.8 ± 1.4 (1.00)
349.7 ± 2.7 (5.40)
342.6 ± 3.3 (5.29)
Solid
Svelte
Hooks
mobX

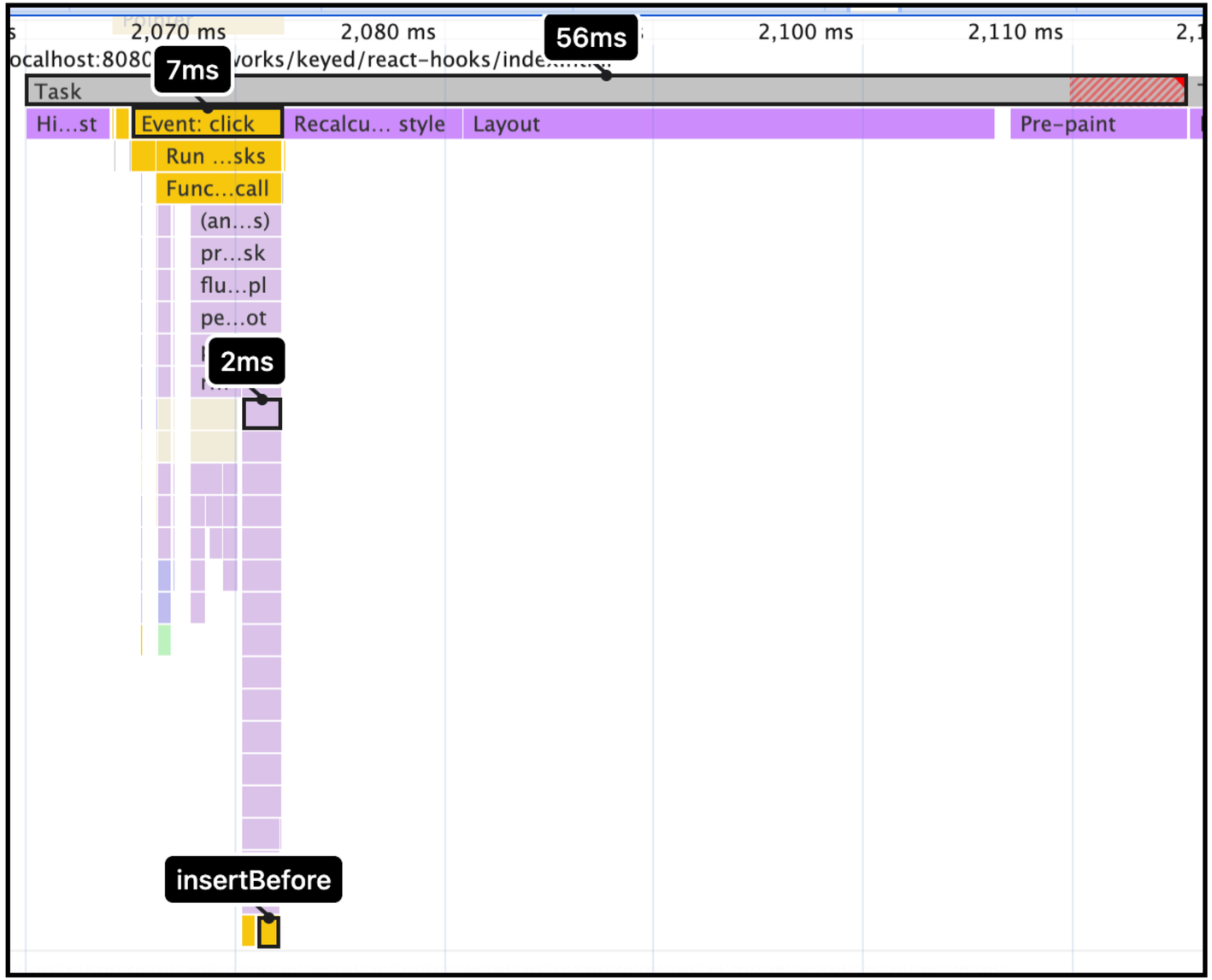
Swap rows (2, 999)



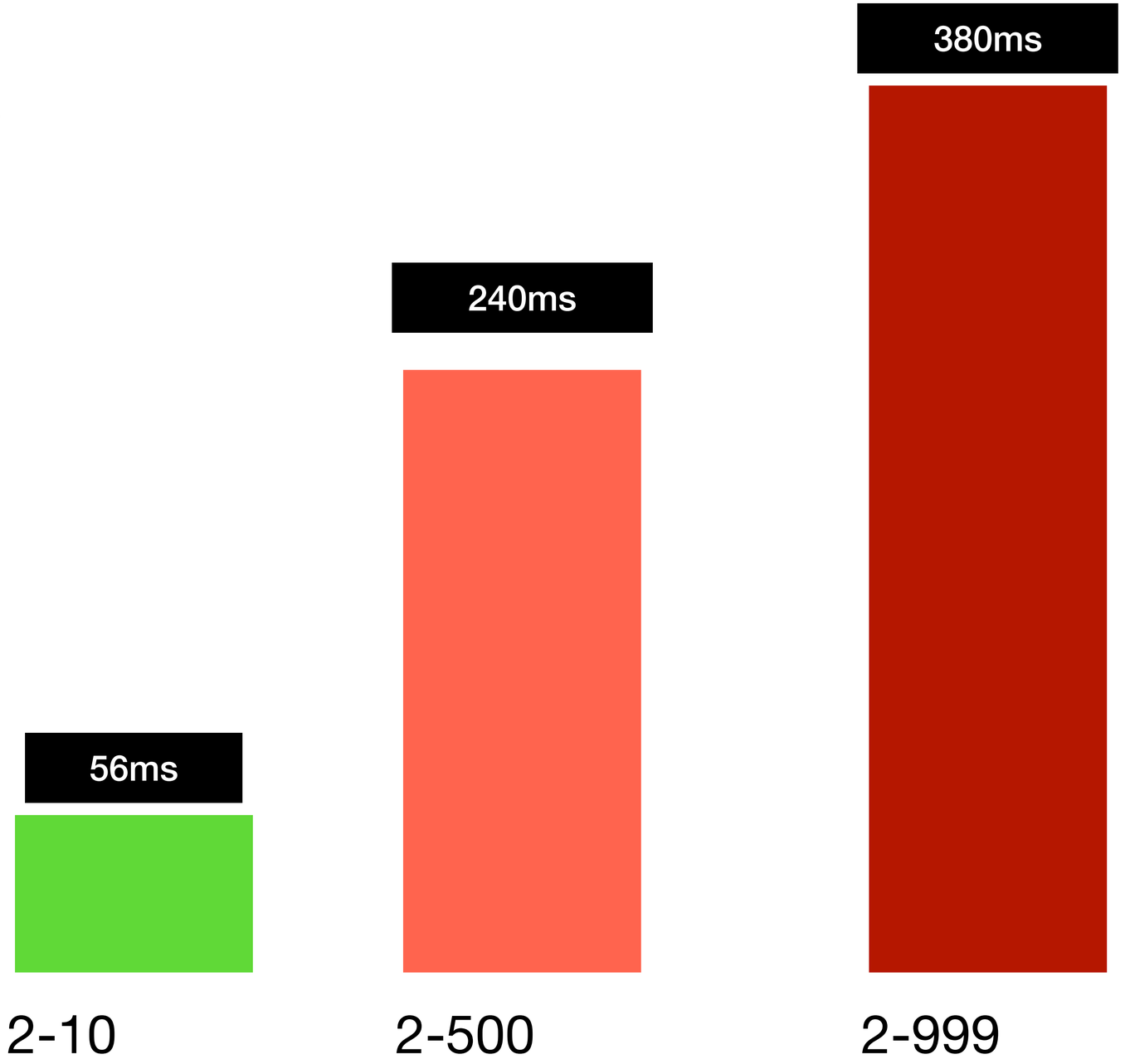
Swap rows (2, 500)



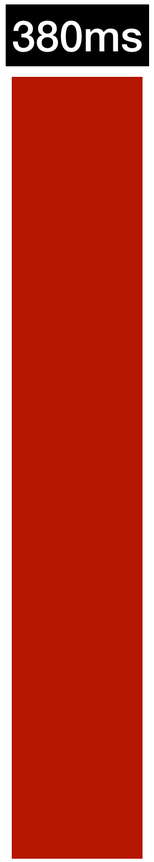
Swap rows (2, 10)



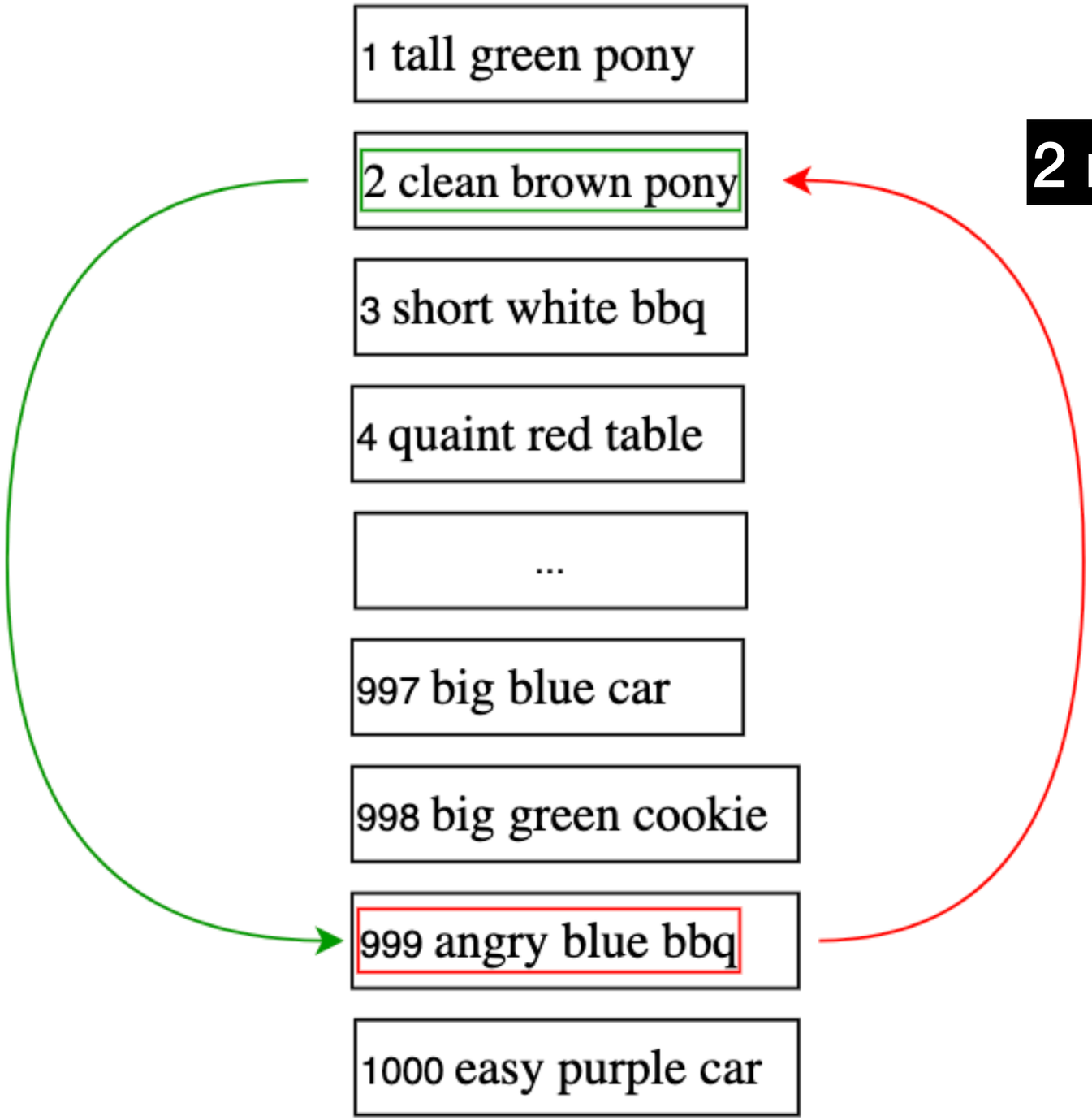
Swap rows



Swap rows



2-999



2 или 999 ?

Swap rows

- https://github.com/facebook/react/blob/8b2e903a7447d370eb77bb117bc4c0ae240ce831/packages/react-reconciler/src/ReactChildFiber.js#L1172

reconcileChildrenArray

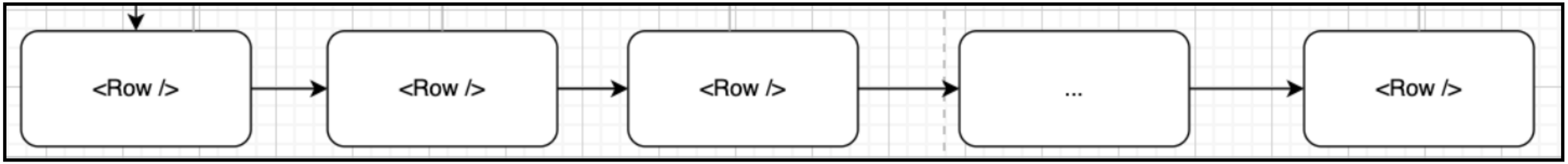
Swap rows (reconcileChildrenArray)

- lastPlacedIndex - до какого элемента новый список совпадает с прошлой версией
- placeChild - на основе lastPlacedIndex и прошлой и текущей позиций элемента определяет необходимо ли «двигать» элемент или его можно оставить на месте. (https://github.com/facebook/react/blob/8b2e903a7447d370eb77bb117bc4c0ae240ce831/packages/react-reconciler/src/ReactChildFiber.js#L511)

Swap rows (reconcileChildrenArray)

- lastPlacedIndex - до какого элемента новый список совпадает с прошлой версией

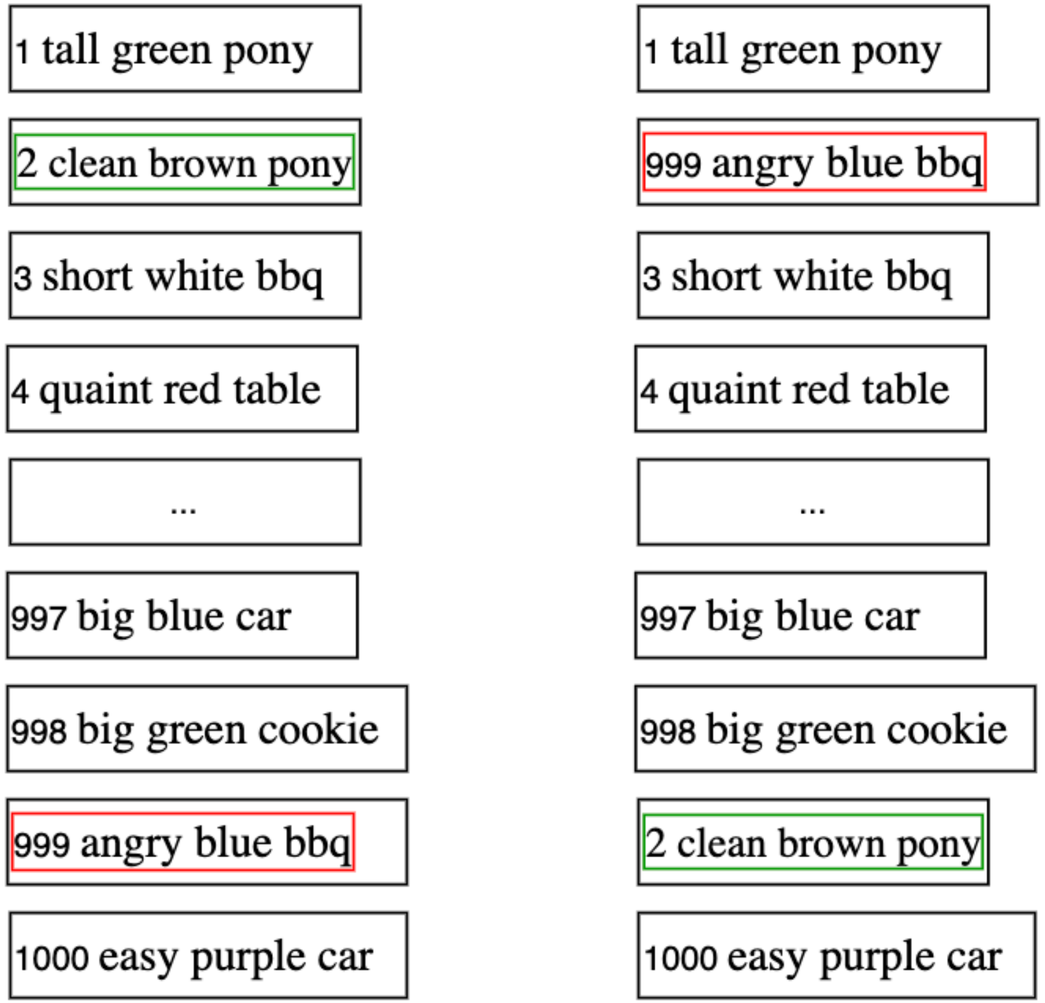
- placeChild - на основе lastPlacedIndex и прошлой и текущей позиций элемента определяет необходимо ли «двигать» элемент или его можно оставить на месте. (https://github.com/facebook/react/blob/8b2e903a7447d370eb77bb117bc4c0ae240ce831/packages/react-reconciler/src/ReactChildFiber.js#L511)



Swap rows (reconcileChildrenArray)

Swap rows (reconcileChildrenArray)

Swap rows (reconcileChildrenArray)



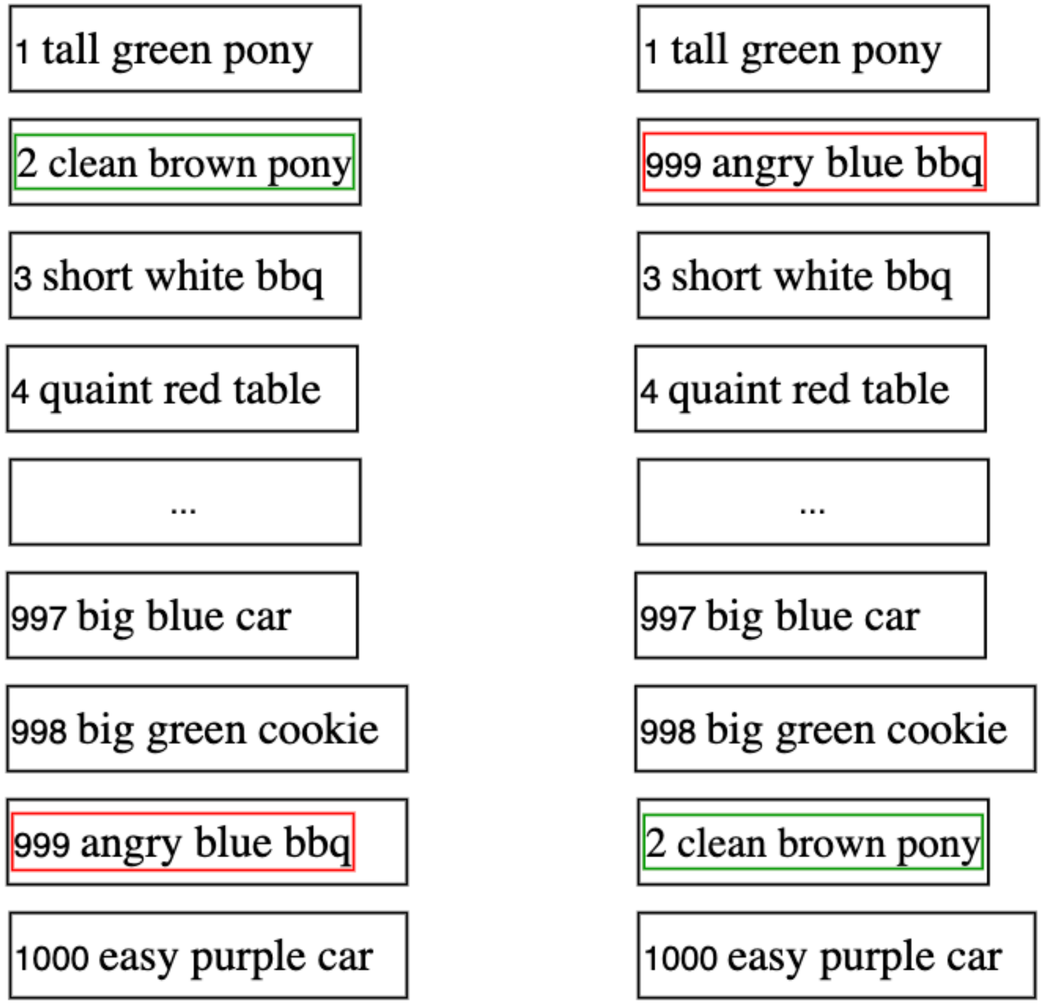

```
const current = newFiber.alternate;
if (current !== null) {
  const oldIndex = current.index;
  if (oldIndex < lastPlacedIndex) {
    // This is a move.
    newFiber.flags |= Placement | PlacementDEV;
    return lastPlacedIndex;
  } else {
    // This item can stay in place.
    return oldIndex;
  }
} else {
  // This is an insertion.
  newFiber.flags |= Placement | PlacementDEV;
  return lastPlacedIndex;
}
```

998

0

placeChild

Swap rows (reconcileChildrenArray)




```
const current = newFiber.alternate;
if (current !== null) {
  co      998      = cu      0      dex;
  if (oldIndex < lastPlacedIndex) {
    // This is a move.
    newFiber.flags |= Placement | PlacementDEV;
    return lastPlacedIndex;
  } else {
    // This item can stay in place.
    return oldIndex;      998
  }
} else {
  // This is an insertion.
  newFiber.flags |= Placement | PlacementDEV;
  return lastPlacedIndex;
}
```

placeChild

```
const current = newFiber.alternate;
if (current !== null) {
  const oldIndex = current.index;
  if (oldIndex < lastPlacedIndex) {
    // This is a move.
    newFiber.flags |= Placement | PlacementDEV;
    return lastPlacedIndex;
  } else {
    // This item can stay in place.
    return oldIndex;
  }
} else {
  // This is an insertion.
  newFiber.flags |= Placement | PlacementDEV;
  return lastPlacedIndex;
}
```

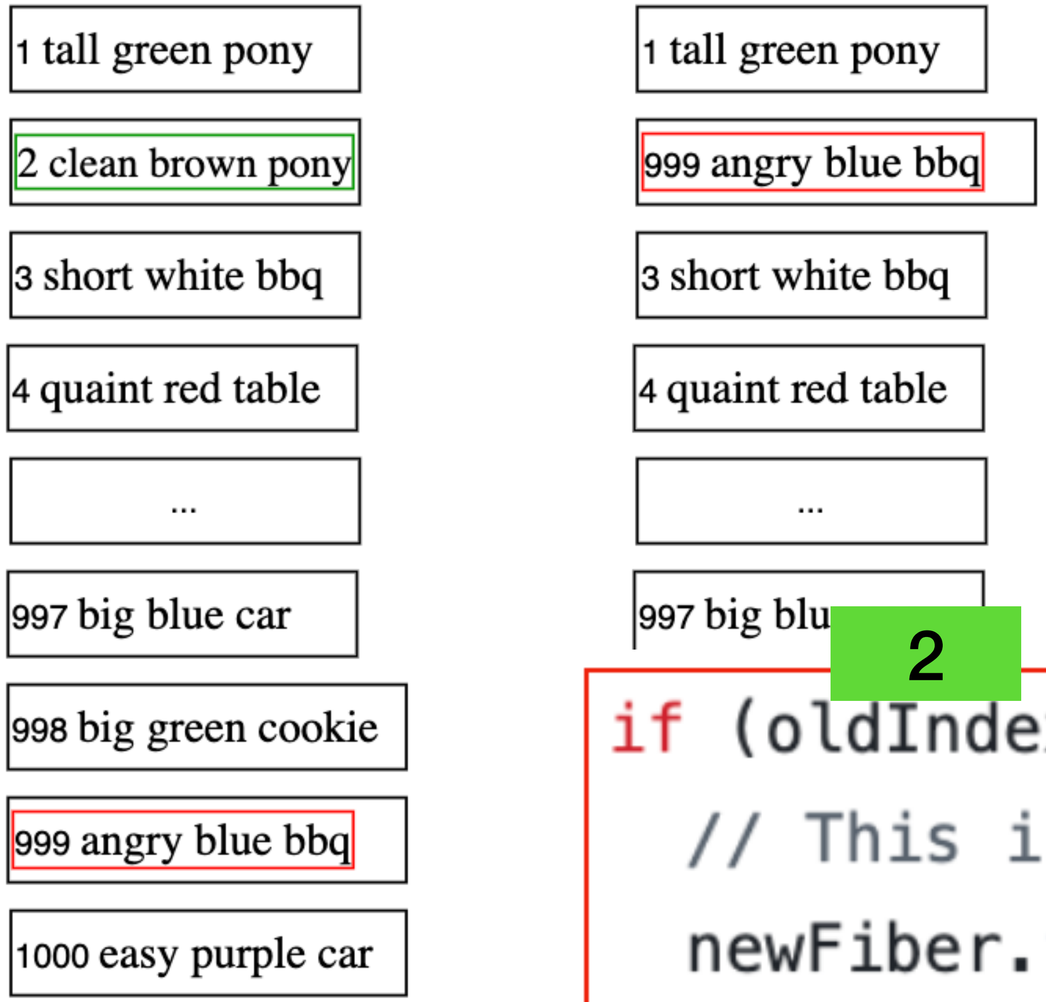
```
const current = newFiber.alternate;
if (current !== null) {
  const oldIndex = current.index;
  if (oldIndex < lastPlacedIndex) {
    // This is a move.
    newFiber.flags |= Placement | PlacementDEV;
    return lastPlacedIndex;
  } else {
    // This item can stay in place.
    return oldIndex;
  }
} else {
  // This is an insertion.
  newFiber.flags |= Placement | PlacementDEV;
  return lastPlacedIndex;
}
```

```
const current = newFiber.alternate;
if (current !== null) {
  const oldIndex = current.index;
  if (oldIndex < lastPlacedIndex) {
    // This is a move.
    newFiber.flags |= Placement | PlacementDEV;
    return lastPlacedIndex;
  } else {
    // This item can stay in place.
    return oldIndex;
  }
} else {
  // This is an insertion.
  newFiber.flags |= Placement | PlacementDEV;
  return lastPlacedIndex;
}
```

```
const current = newFiber.alternate;
if (current !== null) {
  const oldIndex = current.index;
  if (oldIndex < lastPlacedIndex) {
    // This is a move.
    newFiber.flags |= Placement | PlacementDEV;
    return lastPlacedIndex;
  } else {
    // This item can stay in place.
    return oldIndex;
  }
} else {
  // This is an insertion.
  newFiber.flags |= Placement | PlacementDEV;
  return lastPlacedIndex;
}
```

```
const current = newFiber.alternate;
if (current !== null) {
  const oldIndex = current.index;
  if (oldIndex < lastPlacedIndex) {
    // This is a move.
    newFiber.flags |= Placement | PlacementDEV;
    return lastPlacedIndex;
  } else {
    // This item can stay in place.
    return oldIndex;
  }
} else {
  // This is an insertion.
  newFiber.flags |= Placement | PlacementDEV;
  return lastPlacedIndex;
}
```

Swap rows (reconcileChildrenArray)



lastPlacedIndex = 998

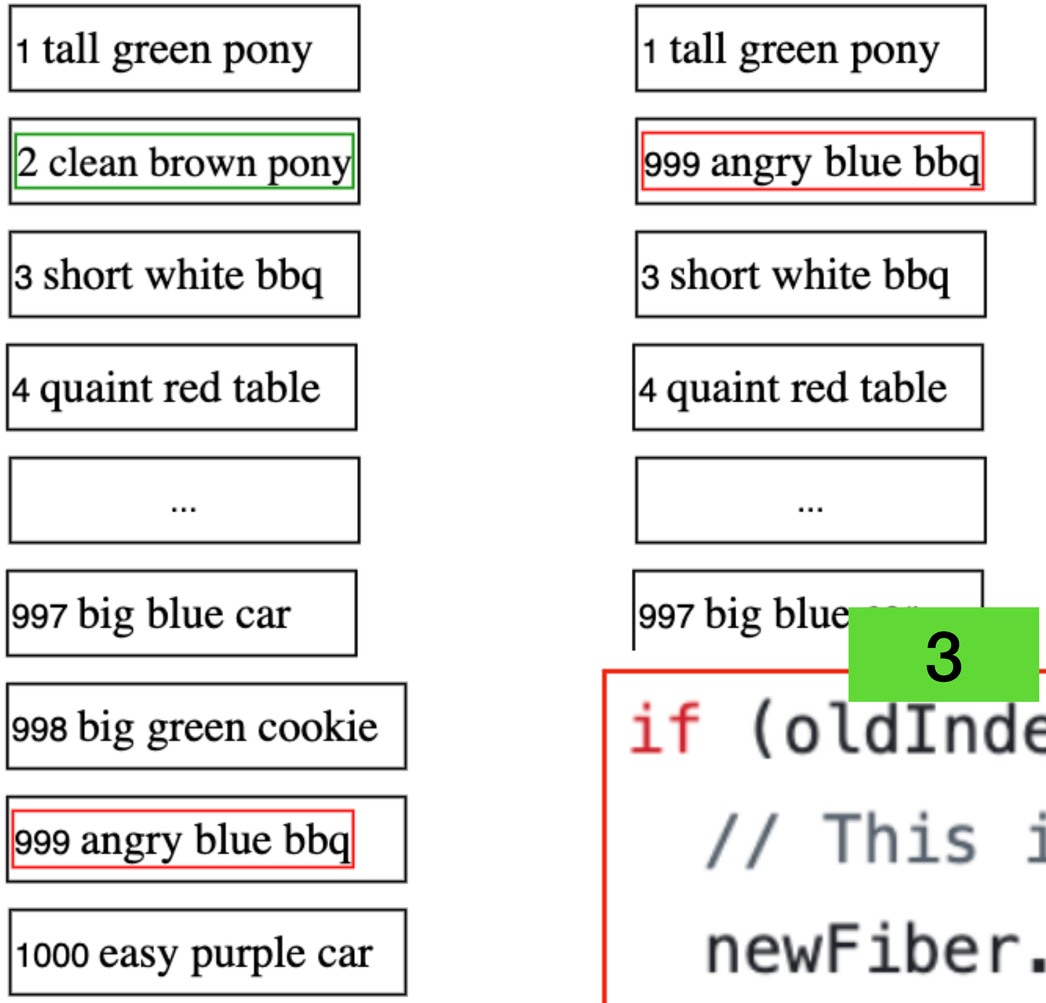
Требуется апдейт в DOM

2

998

```
if (oldIndex < lastPlacedIndex) {
  // This is a move.
  newFiber.flags |= Placement | PlacementDEV;
  return lastPlacedIndex;
```

Swap rows (reconcileChildrenArray)



lastPlacedIndex = 998

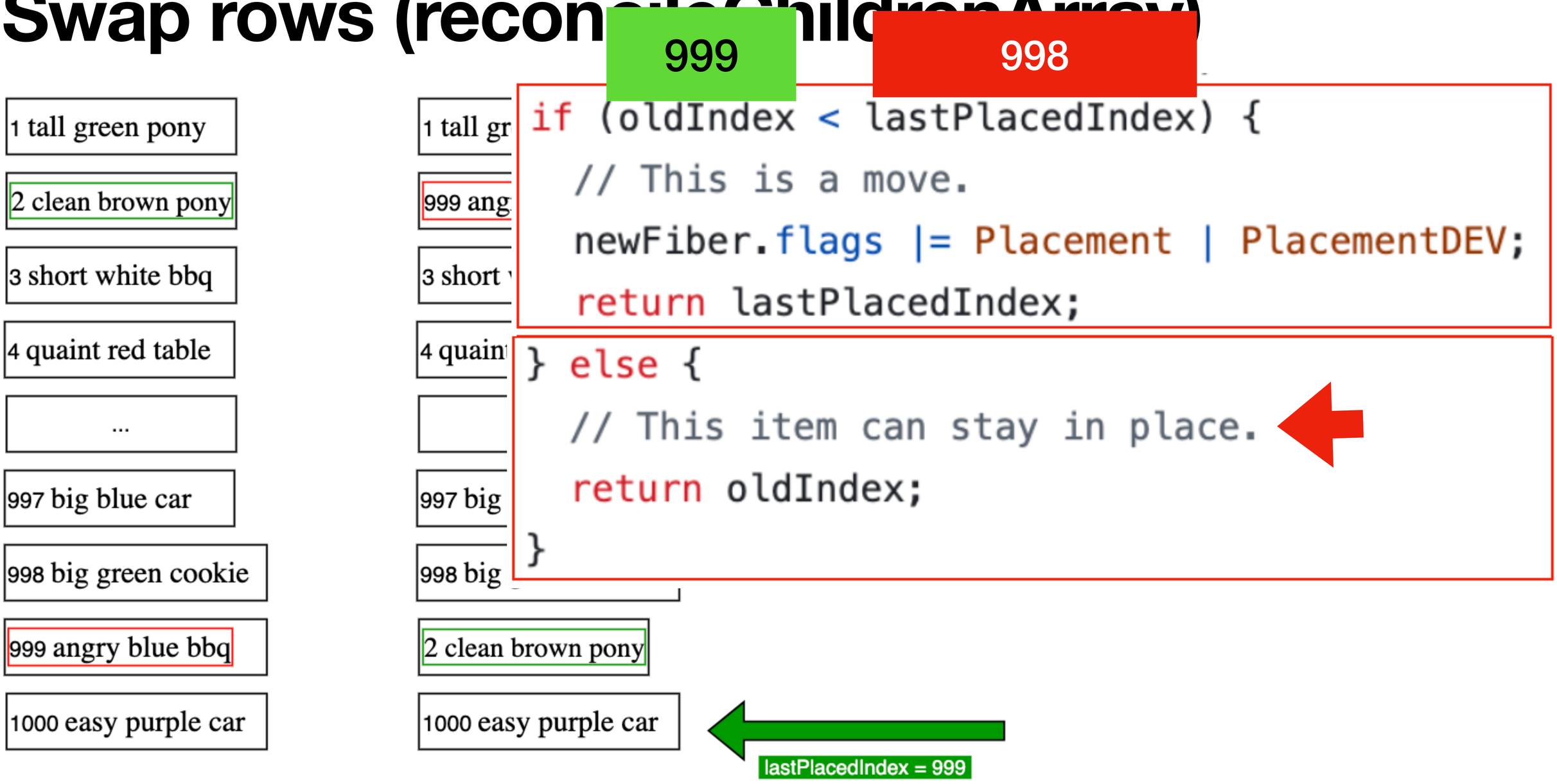
Требуется апдейт в DOM

3

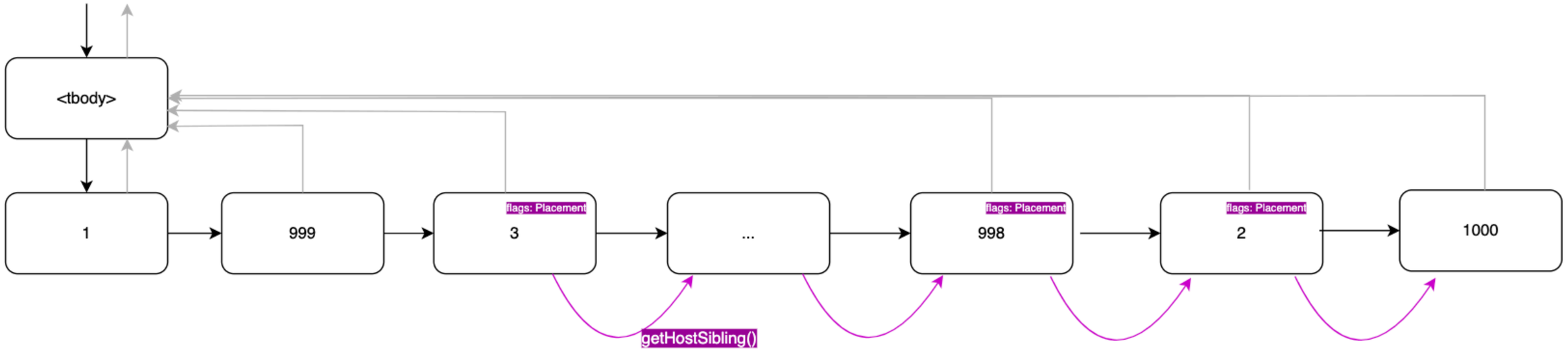
998

```
if (oldIndex < lastPlacedIndex) {
  // This is a move.
  newFiber.flags |= Placement | PlacementDEV;
  return lastPlacedIndex;
```

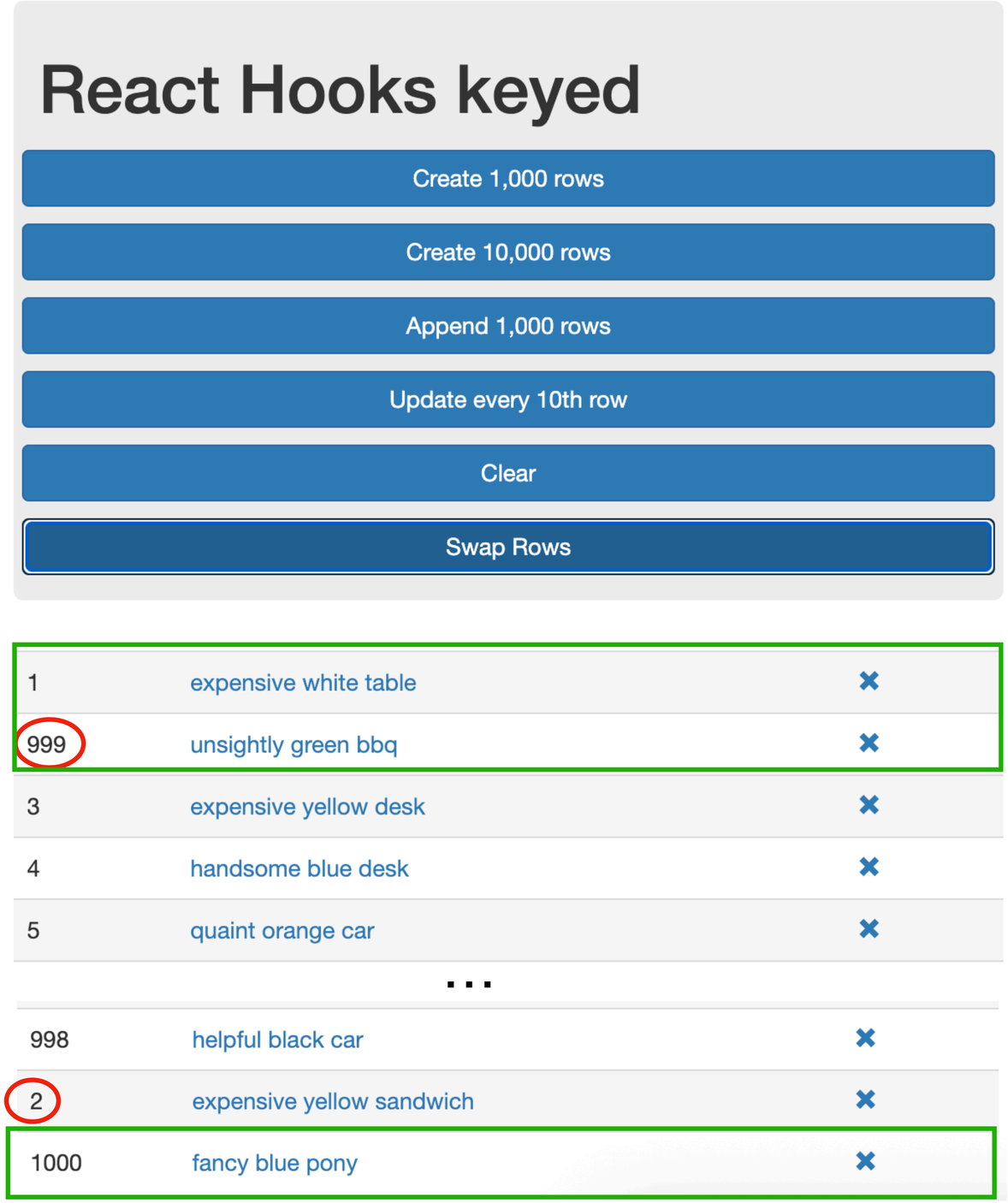
Swap rows (reconcileChildrenArray)



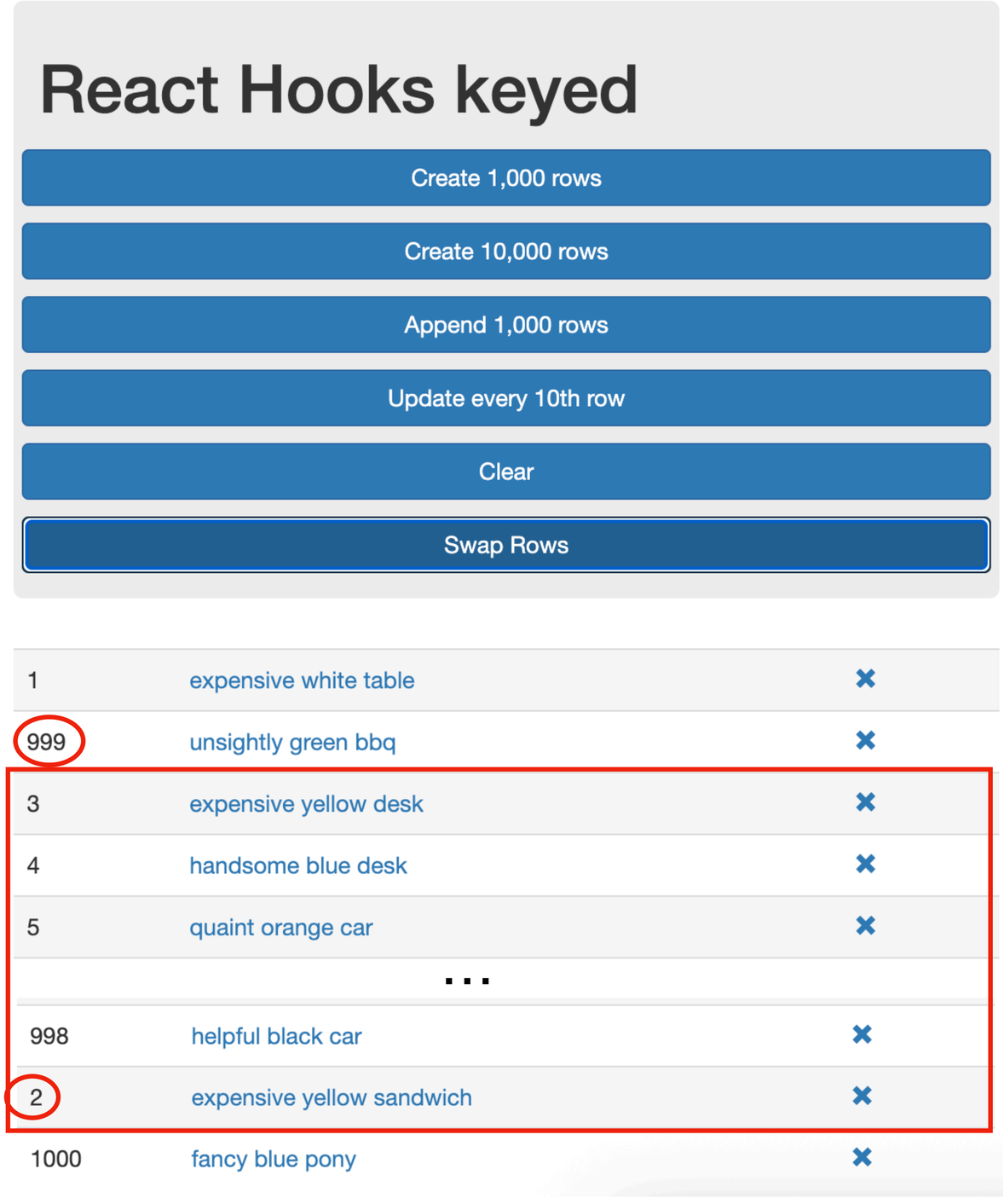
Swap rows (flag Placement)



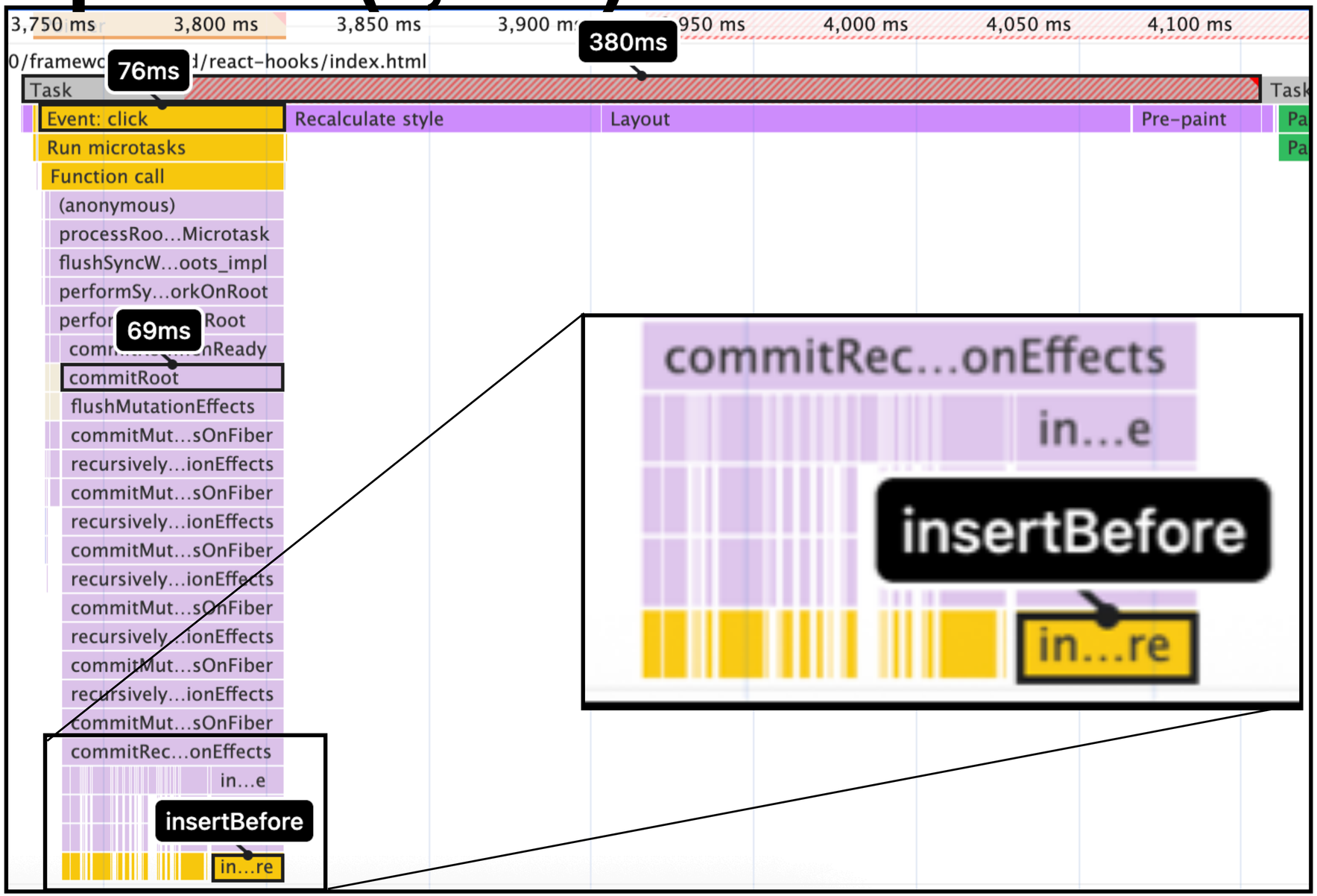
Swap rows


React Hooks keyed
Create 1,000 rows
Create 10,000 rows
Append 1,000 rows
Update every 10th row
Clear
Swap Rows
1 expensive white table
999 unsightly green bbq
3 expensive yellow desk
4 handsome blue desk
5 quaint orange car
998 helpful black car
2 expensive yellow sandwich
1000 fancy blue pony

Swap rows


React Hooks keyed
Create 1,000 rows
Create 10,000 rows
Append 1,000 rows
Update every 10th row
Clear
Swap Rows
1 expensive white table
999 unsightly green bbq
3 expensive yellow desk
4 handsome blue desk
5 quaint orange car
998 helpful black car
2 expensive yellow sandwich
1000 fancy blue pony

Swap rows (2, 999)



Swap rows

Необходимо избежать резкого скачка lastPlacedIndex при обработке элемента с индексом 998

Swap rows

```
const current = newFiber.alternate;
if (current !== null) {
  const oldIndex = current.index;
  if (oldIndex < lastPlacedIndex) {
    // This is a move.
    newFiber.flags |= Placement | PlacementDEV;
    return lastPlacedIndex;
  } else {
    // This item can stay in place.
    return oldIndex;
  }
} else {
  // This is an insertion.
  newFiber.flags |= Placement | PlacementDEV;
  return lastPlacedIndex;
}
```

998

998

0

placeChild

Swap rows

1 tall green pony	1 tall green pony
2 clean brown pony	999 angry blue bbq
3 short white bbq	3 short white bbq
4 quaint red table	4 quaint red table
...	...
997 big blue car	997 big blue car
998 big green cookie	998 big green cookie
999 angry blue bbq	2 clean brown pony
1000 easy purple car	1000 easy purple car

lastPlacedIndex = 0

```
} else {
  // This is an insertion.
  newFiber.flags |= Placement | PlacementDEV;
  return lastPlacedIndex;
}
```

Swap rows

1 tall green pony	1 tall green pony
2 clean brown pony	999 angry blue bbq
3 short white bbq	3 short white bbq
4 quaint red table	4 quaint red table
...	...
997 big blue car	997 big blue car
998 big green cookie	998 big green cookie
999 angry blue bbq	2 clean brown pony
1000 easy purple car	1000 easy purple car

lastPlacedIndex = 0

lastPlacedIndex = 2

```
} else {
  // This item can stay in place.
  return oldIndex;
}
```

Swap rows

1 tall green pony	1 tall green pony
2 clean brown pony	999 angry blue bbq
3 short white bbq	3 short white bbq
4 quaint red table	4 quaint red table
...	...
997 big blue car	997 big blue car
998 big green cookie	998 big green cookie
999 angry blue bbq	2 clean brown pony
1000 easy purple car	1000 easy purple car

lastPlacedIndex = 0

lastPlacedIndex = 2

lastPlacedIndex = 3

```
} else {
  // This item can stay in place.
  return oldIndex;
}
```

Swap rows

1 tall green pony	1 tall green pony
2 clean brown pony	999 angry blue bbq
3 short white bbq	3 short white bbq
4 quaint red table	4 quaint red table
...	...
997 big blue car	997 big blue car
998 big green cookie	998 big green cookie
999 angry blue bbq	2 clean brown pony
1000 easy purple car	1000 easy purple car

lastPlacedIndex = 0

lastPlacedIndex = 2

lastPlacedIndex = 3

lastPlacedIndex = 996

```
} else {
  // This item can stay in place.
  return oldIndex;
}
```

Swap rows

1 tall green pony	1 tall green pony
2 clean brown pony	999 angry blue bbq
3 short white bbq	3 short white bbq
4 quaint red table	4 quaint red table
...	...
997 big blue car	997 big blue car
998 big green cookie	998 big green cookie
999 angry blue bbq	2 clean brown pony
1000 easy purple car	1000 easy purple car

lastPlacedIndex = 0

lastPlacedIndex = 2

lastPlacedIndex = 3

lastPlacedIndex = 996

lastPlacedIndex = 997

```
} else {
  // This item can stay in place.
  return oldIndex;
}
```

Swap rows

1 tall green pony
2 clean brown pony
3 short white bbq
4 quaint red table
...
997 big blue car
998 big green cookie
999 angry blue bbq
1000 easy purple car

1 tall green pony
999 angry blue bl
3 short white bbq
4 quaint red table
...
997 big blue car
998 big green cookie
2 clean brown pony
1000 easy purple car

```
} else {
  // This is an insertion.
  newFiber.flags |= Placement | PlacementDEV;
  return lastPlacedIndex;
}
```

lastPlacedIndex = 3

lastPlacedIndex = 996

lastPlacedIndex = 997

lastPlacedIndex = 997

Swap rows

```
} else {
  // This item can stay in place.
  return oldIndex;
}
```

1 tall green pony	1 tall green pony	lastPlacedIndex = 0
2 clean brown pony	999 angry blue bbq	lastPlacedIndex = 2
3 short white bbq	3 short white bbq	lastPlacedIndex = 3
4 quaint red table	4 quaint red table	
...	...	
997 big blue car	997 big blue car	lastPlacedIndex = 996
998 big green cookie	998 big green cookie	lastPlacedIndex = 997
999 angry blue bbq	2 clean brown pony	lastPlacedIndex = 997
1000 easy purple car	1000 easy purple car	lastPlacedIndex = 999

Swap rows

	Solid	Svelte	Hooks	mobX
slowdown.				
select row highlighting a selected row. (5 warmup runs). 4 x CPU slowdown.				
swap rows swap 2 rows for table with 1,000 rows. (5 warmup runs). 4 x CPU slowdown.	65.4 ± 0.9 (1.05)	64.8 ± 1.4 (1.04)	62.4 ± 23.6 (1.00)	63.5 ± 0.6 (1.02)
remove row				

Swap rows (Svelte)

- https://github.com/sveltejs/svelte/blob/db69b7e345090edd4aa7949c52478adc05a61d61/packages/svelte/src/internal/client/dom/blocks/each.js#L421

```
/** @type {Effect[]} */
var matched = [];

/** @type {Effect[]} */
var stashed = [];
```

```
{#each data as row (row)}
```

```
if (matched.length < stashed.length) {
  // more efficient to move later items to the front
```

```
} else {
  // more efficient to move earlier items to the back
  seen.delete(effect);
  move(effect, current, anchor);
```

Swap rows (Solid)

```
<For each={data()}>
```

- https://github.com/ryansolid/dom-expressions/blob/96bce81c8468ea951cd5a943838b9621975bf906/packages/dom-expressions/src/reconcile.js#L2

```
// swap backward
} else if (a[aStart] === b[bEnd - 1] && b[bStart] === a[aEnd - 1]) {
  const node = a[--aEnd].nextSibling;
  parentNode.insertBefore(b[bStart++], a[aStart++].nextSibling);
  parentNode.insertBefore(b[--bEnd], node);

  a[aEnd] = b[bEnd];
```

Swap rows

```
{data.map(item => (
  <Row key={item.id} item={item} selected={selected === item.id} dispatch={dispatch} />
))}
```

Swap rows

Svelte/Solid

Diff в массиве

React

Diff в однонаправленном связном списке

Swap rows

Closed **fix exponential search in getHostSibling function** #22697

guojin-long wants to merge 2 commits into `facebook:main` from `guojin-long:main`

gaearon commented on Nov 6, 2021

Do we have a sense of how much overhead this adds for the common case?

swap rows swap 2 rows for table with 1,000 rows. (5 warmup runs). 4x CPU slowdown.	101.4 ± 2.5 (1.00)	801.7 ± 14.1 (7.91)

guojin-long closed this on Nov 25, 2022

https://github.com/facebook/react/pull/22697

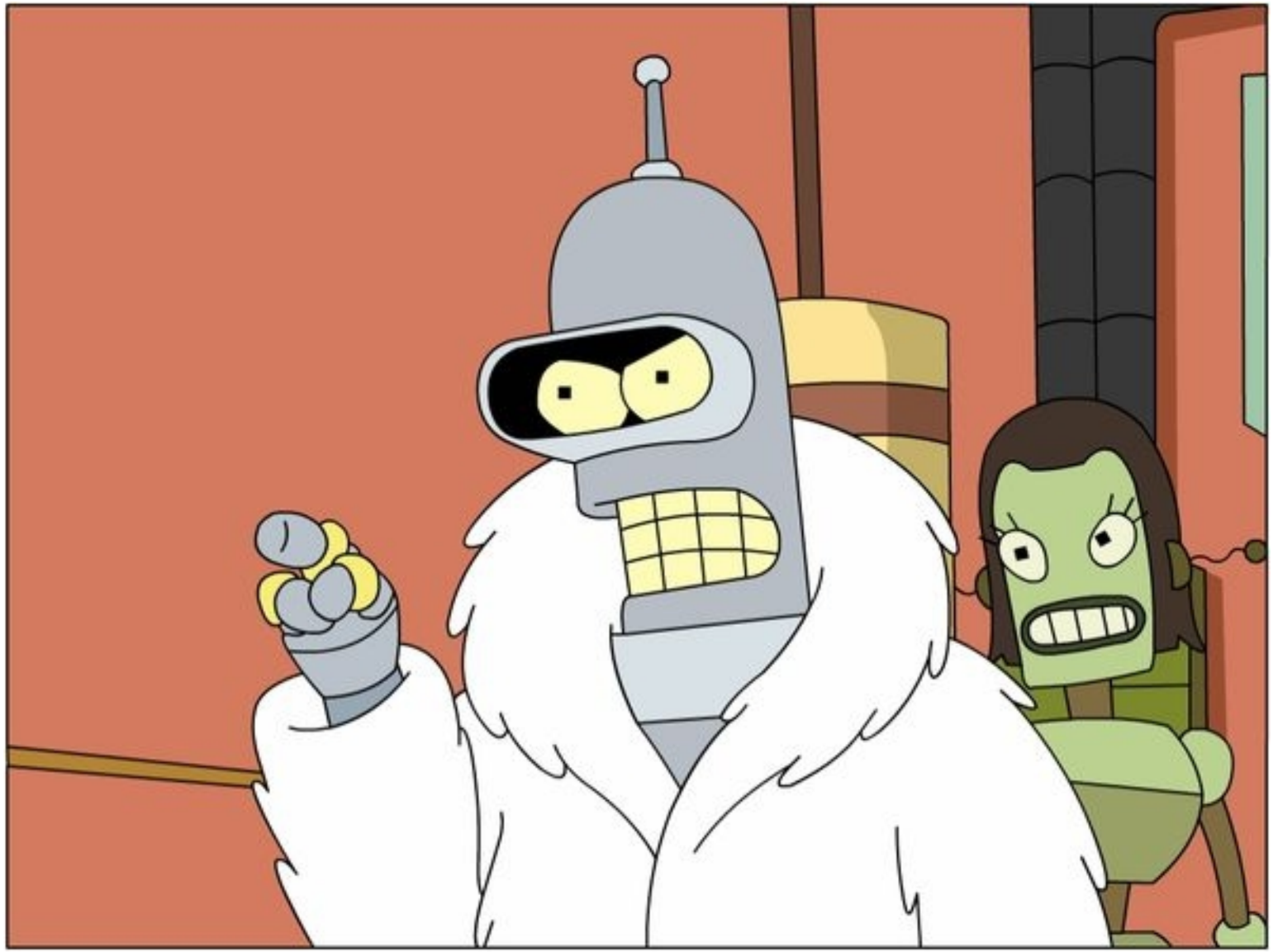

Можно,а зачем?

trueadm commented on Mar 23

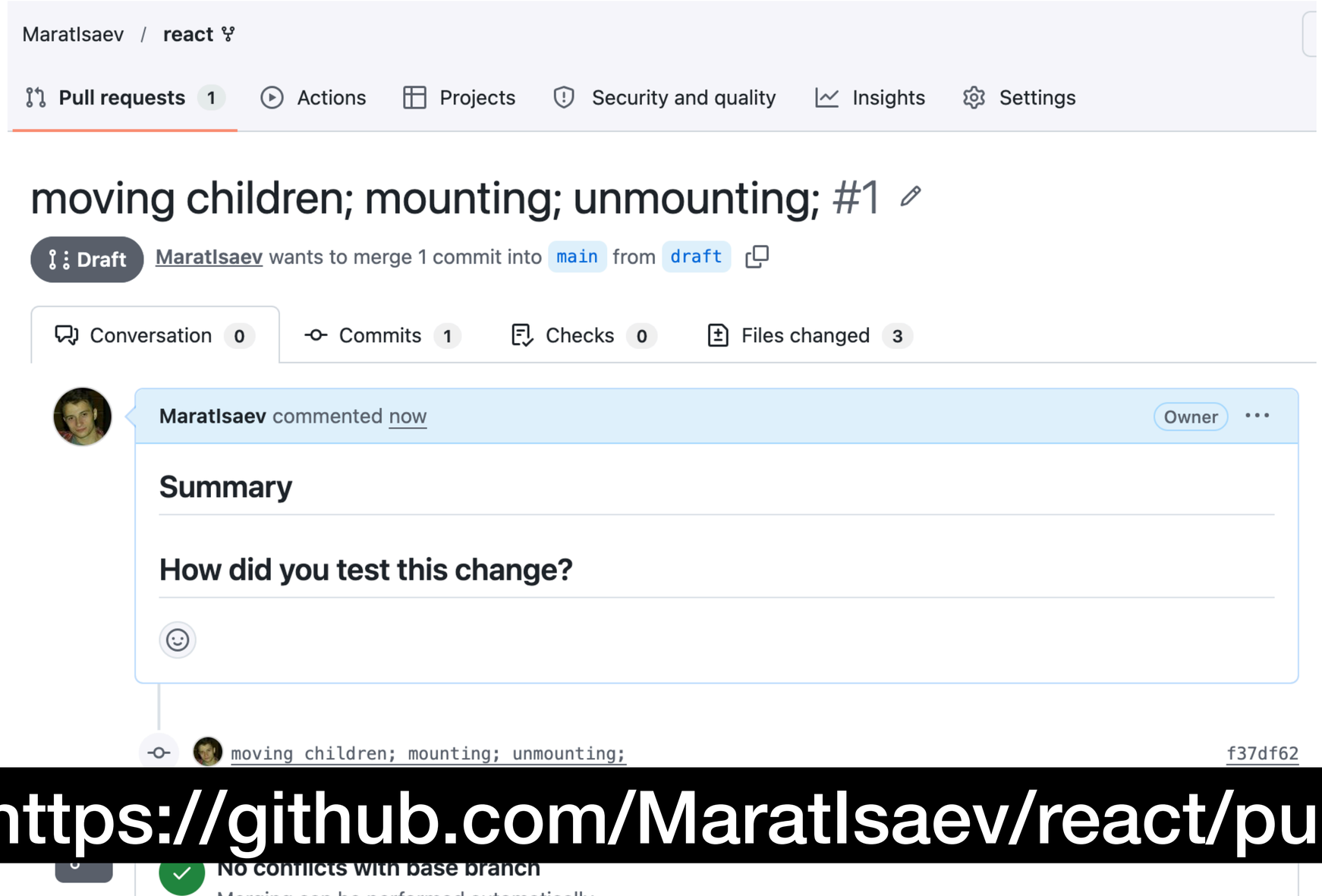
Removing the `tbody` element means you're doing something mechanically different to what this benchmark is meant to show. This is particularly dangerous as having an empty `<table>` HTML element means browsers will automatically insert `tbody` element if one is not there – and that heuristic is important for tables.

The correct thing to do here is fix this in React's reconciliation logic, rather than gaming a benchmark to make React work more effectively with this UI pattern.

Мы построим свой React

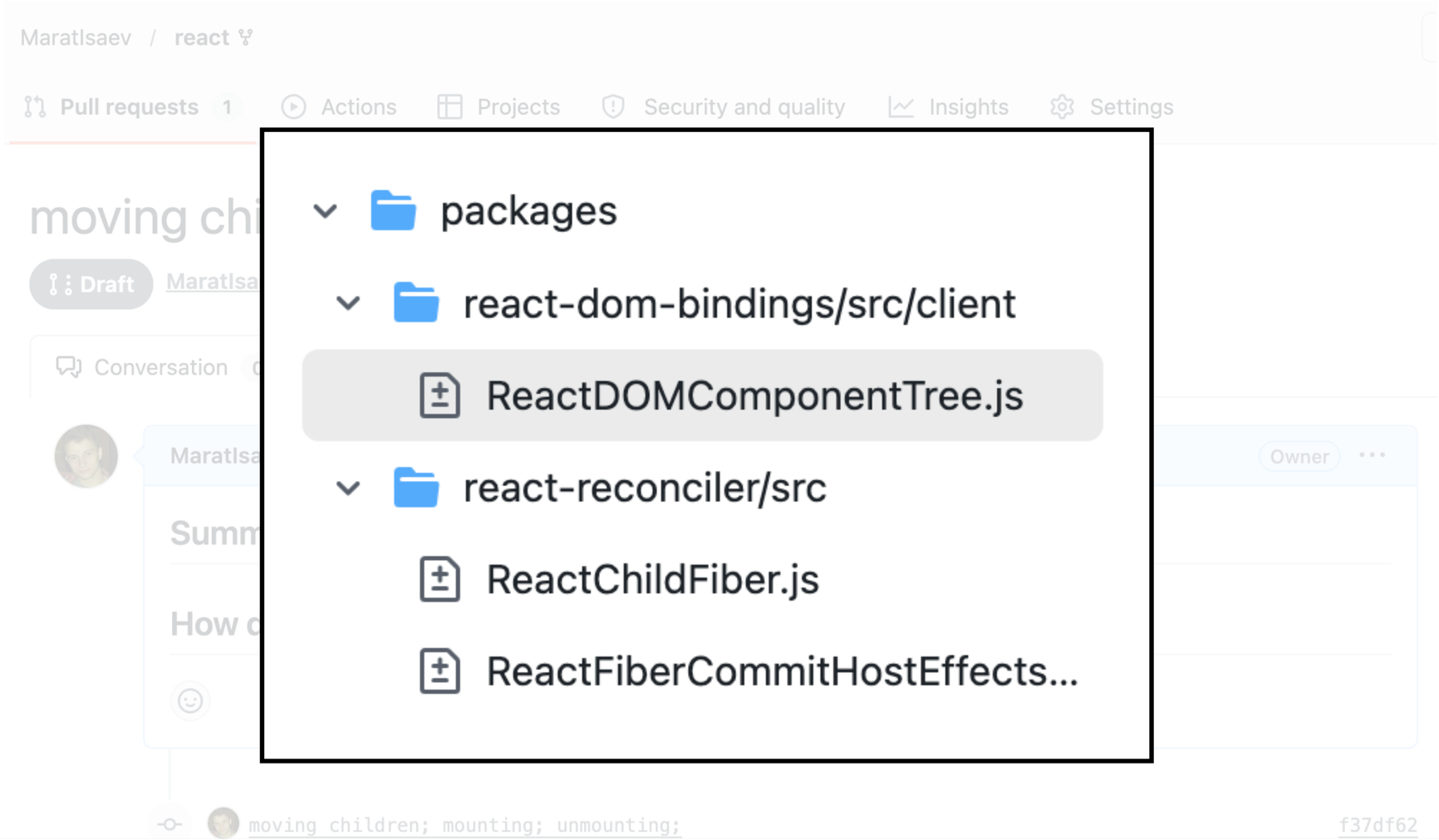


Наш pull request



https://github.com/MaratIsaev/react/pull/1

Haш pull request

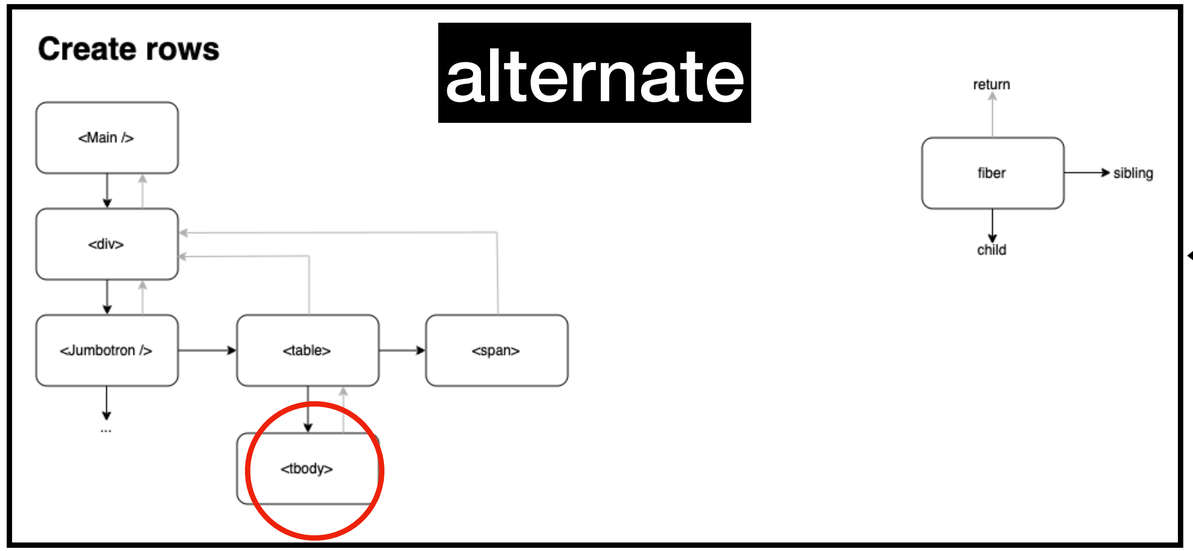


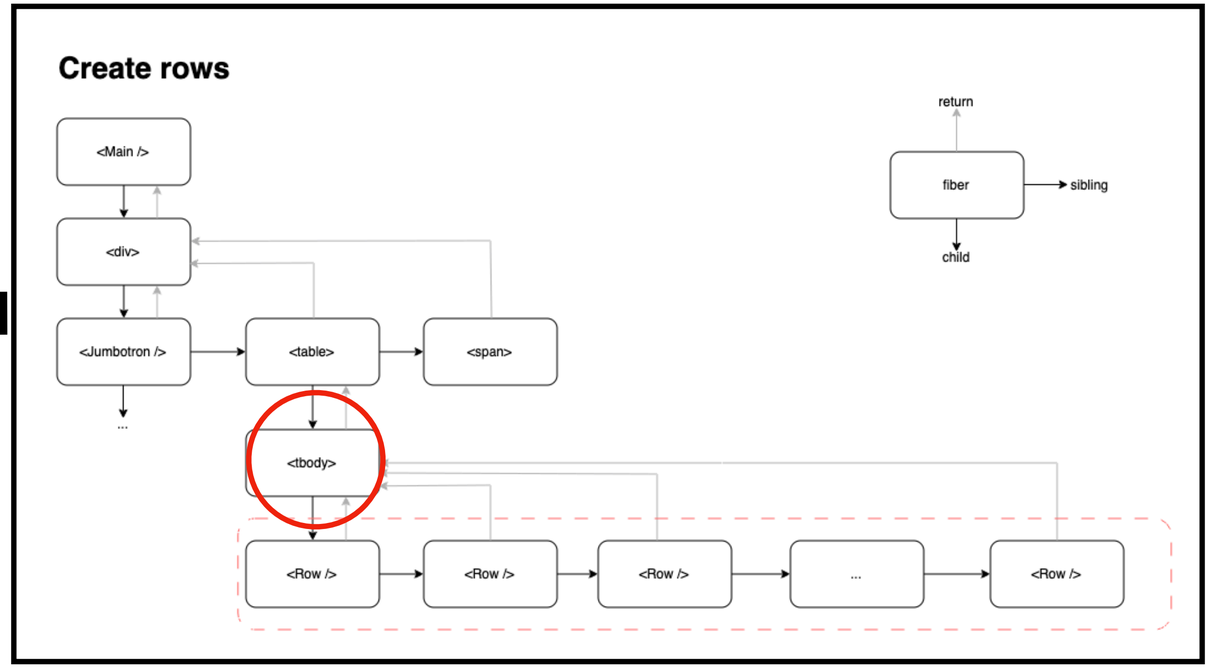
https://github.com/MaratIsaev/react/pull/1

Create (many) rows

```
326  326    function getHostSibling(fiber: Fiber): ?Instance {
327  327      // We're going to search forward into the tree until we find a sibling host
328  328      // node. Unfortunately, if multiple insertions are done in a row we have to
329  329      // search past them. This leads to exponential search for the next sibling.
330  330      // TODO: Find a more efficient way to do this.
331  331      let node: Fiber = fiber;
332  332      siblings: while (true) {
     333  +     if (
     334  +       node.return !== null &&
     335  +       node.sibling !== null &&
     336  +       node.sibling.return === node.return &&
     337  +       node.return.alternate !== null &&
     338  +       node.return.alternate.child === null
     339  +     ) {
     340  +       if (isHostParent(node.return)) {
     341  +         return null;
     342  +       } else {
     343  +         node = node.return;
     344  +         continue siblings;
     345  +       }
     346  +     }
333  347        // If we didn't find anything, let's try the next sibling.
334  348        while (node.sibling === null) {
```

Create (many) rows





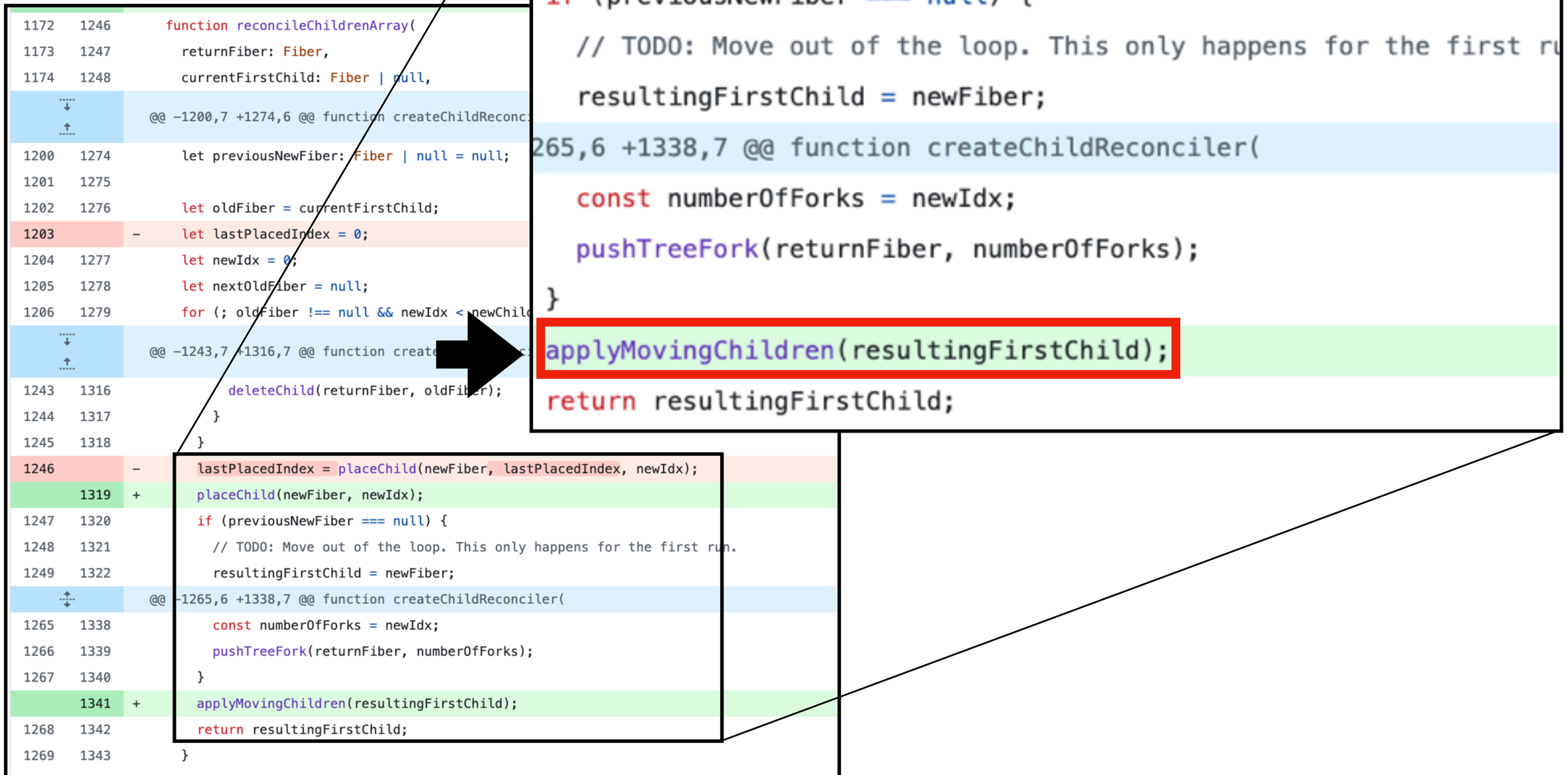
Swap rows

```
function placeChild(
  newFiber: Fiber,
  lastPlacedIndex: number,
  newIndex: number,
): number {
function placeChild(newFiber: Fiber, newIndex: number) {
```

```
511     -  function placeChild(
512     -    newFiber: Fiber,
513     -    lastPlacedIndex: number,
514     -    newIndex: number,
515     -  ): number {
        511 +  function placeChild(newFiber: Fiber, newIndex: number) {
516 512        newFiber.index = newIndex;
517 513        if (!shouldTrackSideEffects) {
518 514          // During hydration, the useId algorithm needs to know which fibers are
519 515          // part of a list of children (arrays, iterators).
520 516          newFiber.flags |= Forked;
521     -        return lastPlacedIndex;
        517 +        return;
            }
            const current = newFiber.alternate;
            if (current !== null) {
              const oldIndex = current.index;
              if (oldIndex < lastPlacedIndex) {
                // This is a move.
                newFiber.flags |= Placement | PlacementDEV;
                return lastPlacedIndex;
              } else {
                // This item can stay in place.
                return oldIndex;
533     -        }
        521 +        // This is a move. Move it in applyMovingChildren.
        522 +        return;
534 523        } else {
535 524          // This is an insertion.
536 525          newFiber.flags |= Placement | PlacementDEV;
537     -        return lastPlacedIndex;
        526 +        return;
538 527        }
```

Keyno

Swap rows


1172 1246 function reconcileChildrenArray(
1173 1247 returnFiber: Fiber,
1174 1248 currentFirstChild: Fiber | null,
@@ -1200,7 +1274,6 @@ function createChildReconc
1200 1274 let previousNewFiber: Fiber | null = null;
1201 1275
1202 1276 let oldFiber = currentFirstChild;
1203 - let lastPlacedIndex = 0;
1204 1277 let newIdx = 0;
1205 1278 let nextOldFiber = null;
1206 1279 for (; oldFiber !== null && newIdx <
@@ -1243,7 +1316,7 @@ function create
1243 1316 deleteChild(returnFiber, oldFiber);
1244 1317 }
1245 1318 }
1246 - lastPlacedIndex = placeChild(newFiber, lastPlacedIndex, newIdx);
1319 + placeChild(newFiber, newIdx);
1247 1320 if (previousNewFiber === null) {
1248 1321 // TODO: Move out of the loop. This only happens for the first run.
1249 1322 resultingFirstChild = newFiber;
@@ -1265,6 +1338,7 @@ function createChildReconciler(
1265 1338 const numberOfForks = newIdx;
1266 1339 pushTreeFork(returnFiber, numberOfForks);
1267 1340 }
1341 + applyMovingChildren(resultingFirstChild);
1268 1342 return resultingFirstChild;
1269 1343 }

lastPlacedIndex = placeChild(newFiber, lastPlacedIndex, newIdx);
placeChild(newFiber, newIdx);
if (previousNewFiber === null) {
// TODO: Move out of the loop. This only happens for the first ru
resultingFirstChild = newFiber;
265,6 +1338,7 @@ function createChildReconciler(
const numberOfForks = newIdx;
pushTreeFork(returnFiber, numberOfForks);
}
applyMovingChildren(resultingFirstChild);
return resultingFirstChild;

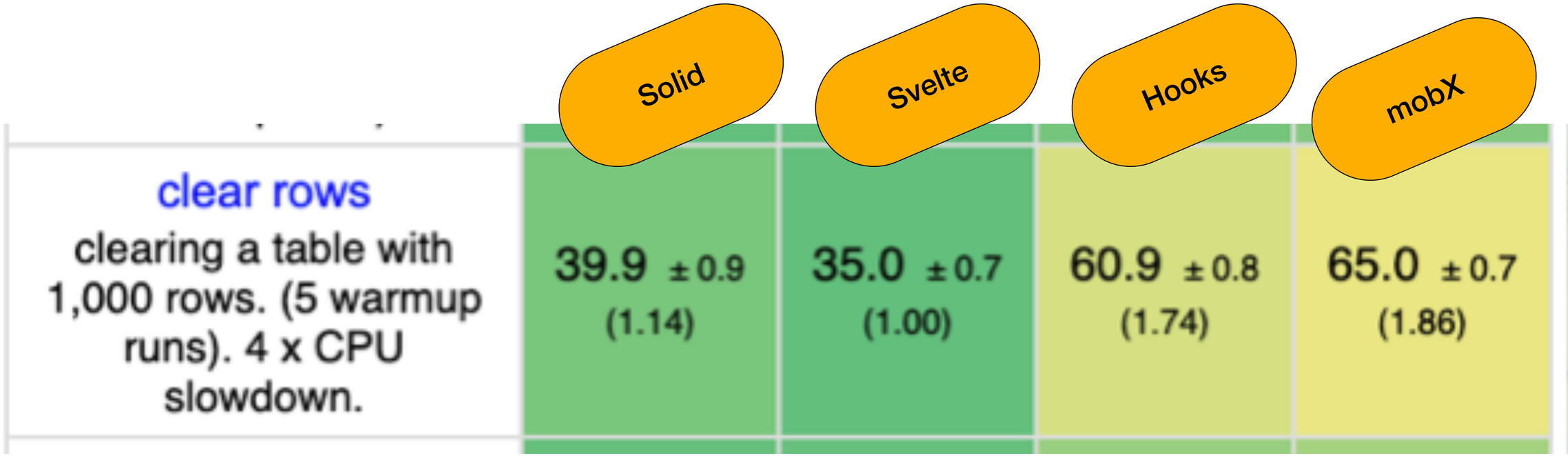
Swap rows

```
1161 +  function applyMovingChildren(resultingFirstChild: Fiber | null) {
1162 +    if (!shouldTrackSideEffects || !resultingFirstChild) {
1163 +      // Noop.
1164 +      return null;
1165 +    }
1166 +
1167 +    let a = [];
1168 +    const b = [];
1169 +
1170 +    let next = resultingFirstChild;
1171 +
1172 +    do {
1173 +      if (!next) {
1174 +        break;
1175 +      } else if (
1176 +        !(next.flags & Placement) &&
1177 +        next.alternate !== null &&
1178 +        next.key !== null
1179 +      ) {
1180 +        if (next.index !== next.alternate.index) {
1181 +          a.push(next.alternate);
1182 +          b.push(next);
1183 +        }
1184 +      }
1185 +      next = next.sibling;
1186 +    } while (true);
1187 +
1188 +    if (!b.length) return null;
1189 +
1190 +    a = a.sort((oldFiberA, oldFiberB) => oldFiberA.index - oldFiberB.index);
1191 +
1192 +    const bLength = b.length;
1193 +    let aEnd = a.length;
```

```
1193 +    let aEnd = a.length;
1194 +    let bEnd = bLength;
1195 +    let aStart = 0;
1196 +    let bStart = 0;
1197 +    let map = null;
1198 +
1199 +    while (aStart < aEnd || bStart < bEnd) {
1200 +      if ((a[aStart] && a[aStart].key) === (b[bStart] && b[bStart].key)) {
1201 +        aStart++;
1202 +        bStart++;
1203 +        continue;
1204 +      }
1205 +      while (
1206 +        (a[aEnd - 1] && a[aEnd - 1].key) === (b[bEnd - 1] && b[bEnd - 1].key)
1207 +      ) {
1208 +        aEnd--;
1209 +        bEnd--;
1210 +      }
1211 +      if (aEnd === aStart) {
1212 +        while (bStart < bEnd) b[bStart++].flags |= Placement | PlacementDEV;
1213 +      } else if (bEnd === bStart) {
1214 +        break;
1215 +      } else {
1216 +        if (!map) {
1217 +          map = new Map();
1218 +          let i = bStart;
1219 +          while (i < bEnd) map.set(b[i].key, i++);
1220 +        }
1221 +
1222 +        const index = a[aStart] && map.get(a[aStart].key);
1223 +        if (bStart < index && index < bEnd) {
1224 +          let i = aStart;
1225 +          let sequence = 1;
1226 +          let t;
```

```
let a = [];

const b = [];
```

Clear rows


Solid
Svelte
Hooks
mobX
clear rows
clearing a table with 1,000 rows. (5 warmup runs). 4 x CPU slowdown.
39.9 ± 0.9
(1.14)
35.0 ± 0.7
(1.00)
60.9 ± 0.8
(1.74)
65.0 ± 0.7
(1.86)

Clear rows

```
70  70    export function detachDeletedInstance(node: Instance): void {
71  71      if (enableInternalInstanceMap) {
72  72        internalInstanceMap.delete(node);
73  73        internalPropsMap.delete(node);
74  74        delete (node: any)[internalEventHandlersKey];
75  75        delete (node: any)[internalEventHandlerListenersKey];
76  76        delete (node: any)[internalEventHandlesSetKey];
77  77        delete (node: any)[internalRootNodeResourcesKey];
78  78        if (__DEV__) {
79  79          delete (node: any)[internalInstanceKey];
80  80        }
81  81        return;
82  82      }
83  83      // TODO: This function is only called on host components. I don't think all of
84  84      // these fields are relevant.
85      -   delete (node: any)[internalInstanceKey];
86      -   delete (node: any)[internalPropsKey];
87      -   delete (node: any)[internalEventHandlersKey];
88      -   delete (node: any)[internalEventHandlerListenersKey];
89      -   delete (node: any)[internalEventHandlesSetKey];
    85  +   // eslint-disable-next-line ft-flow/no-unused-expressions
    86  +   (node: any)[internalInstanceKey] && (node[internalInstanceKey] = undefined);
    87  +   // eslint-disable-next-line ft-flow/no-unused-expressions
    88  +   (node: any)[internalPropsKey] && (node[internalPropsKey] = undefined);
    89  +   // eslint-disable-next-line ft-flow/no-unused-expressions
    90  +   (node: any)[internalEventHandlersKey] &&
    91  +     (node[internalEventHandlersKey] = undefined);
    92  +   // eslint-disable-next-line ft-flow/no-unused-expressions
    93  +   (node: any)[internalEventHandlerListenersKey] &&
    94  +     (node[internalEventHandlerListenersKey] = undefined);
    95  +   // eslint-disable-next-line ft-flow/no-unused-expressions
    96  +   (node: any)[internalEventHandlesSetKey] &&
    97  +     (node[internalEventHandlesSetKey] = undefined);
90  98    }
```

&&

```
(node: any)[internalInstanceKey] && (node[internalInstanceKey] = undefined);
```

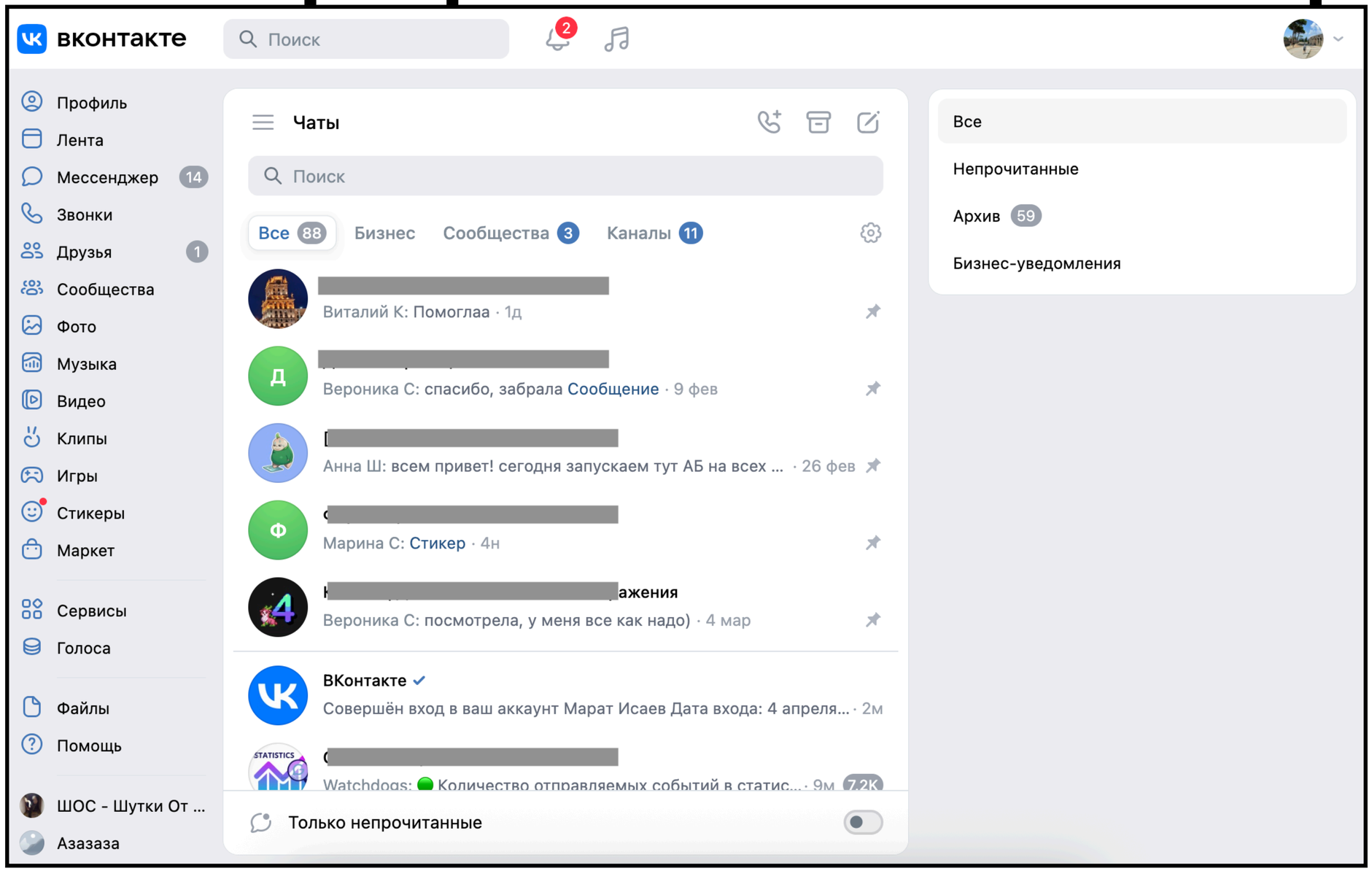
Результат

Name Duration for...	svelte-v5.42.1	solid-v1.9.3	react-hooks-v19.2.0	react-mobX-v19.0.0 + 6.13.5
Implementation notes				
Implementation link	code	code	code	code
create rows creating 1,000 rows. (5 warmup runs).	90.3 ± 1.0 (1.02)	88.9 ± 0.8 (1.00)	103.5 ± 1.2 (1.16)	113.2 ± 0.8 (1.27)
replace all rows updating all 1,000 rows. (5 warmup runs).	97.3 ± 0.9 (1.01)	96.5 ± 0.9 (1.00)	111.9 ± 1.3 (1.16)	121.8 ± 1.3 (1.26)
partial update updating every 10th row for 1,000 row. (3 warmup runs). 4 x CPU slowdown.	52.0 ± 1.2 (1.01)	51.7 ± 0.4 (1.00)	61.2 ± 1.5 (1.18)	66.1 ± 1.1 (1.28)
select row highlighting a selected row. (5 warmup runs). 4 x CPU slowdown.	18.8 ± 0.9 (1.21)	15.6 ± 1.0 (1.00)	18.2 ± 0.8 (1.17)	21.2 ± 0.8 (1.36)
swap rows swap 2 rows for table with 1,000 rows. (5 warmup runs). 4 x CPU slowdown.	58.7 ± 0.8 (1.00)	59.9 ± 0.8 (1.02)	62.4 ± 1.9 (1.06)	63.3 ± 0.7 (1.08)
remove row removing one row. (5 warmup runs). 2 x CPU slowdown.	45.6 ± 0.7 (1.04)	46.5 ± 0.7 (1.06)	44.0 ± 0.5 (1.00)	49.5 ± 0.6 (1.13)
create many rows creating 10,000 rows. (5 warmup runs).	906.6 ± 5.5 (1.01)	901.9 ± 4.5 (1.00)	1,004.3 ± 13.8 (1.11)	1,094.7 ± 2.5 (1.21)
append rows to large table appending 1,000 to a table of 1,000 rows. (5 warmup runs).	107.7 ± 0.8 (1.00)	108.2 ± 3.3 (1.00)	125.1 ± 2.4 (1.16)	131.8 ± 2.6 (1.22)
clear rows clearing a table with 1,000 rows. (5 warmup runs). 4 x CPU slowdown.	34.3 ± 0.8 (1.00)	39.0 ± 0.7 (1.14)	41.3 ± 0.8 (1.20)	42.9 ± 1.2 (1.25)
weighted geometric mean of all factors in the table	1.02	1.02	1.14	1.23

Результат

Name Duration for...	svelte-v5.42.1	solid-v1.9.3	react-hooks-v19.2.0	react-mobX-v19.0.0 + 6.13.5
Implementation notes				
Implementation link	code	code	code	code
create rows creating 1,000 rows. (5 warmup runs).	90.3 ±1.0 (1.02)	88.9 ±0.8 (1.00)	103.5 ±1.2 (1.16)	113.2 ±0.8 (1.27)
replace all rows updating all 1,000 rows. (5 warmup runs).	97.3 ±0.9 (1.01)	96.5 ±0.9 (1.00)	111.9 ±1.3 (1.16)	121.8 ±1.3 (1.26)
partial update updating every 10th row for 1,000 row. (3 warmup runs). 4 x CPU slowdown.	52.0 ±1.2 (1.01)	51.7 ±0.4 (1.00)	61.2 ±1.5 (1.18)	66.1 ±1.1 (1.28)
select row highlighting a selected row. (5 warmup runs). 4 x CPU slowdown.	18.8 ±0.9 (1.21)	15.6 ±1.0 (1.00)	18.2 ±0.8 (1.17)	21.2 ±0.8 (1.36)
swap rows swap 2 rows for table with 1,000 rows. (5 warmup runs). 4 x CPU slowdown.	58.7 ±0.8 (1.00)	59.9 ±0.8 (1.02)	62.4 ±1.9 (1.06)	63.3 ±0.7 (1.08)
remove row removing one row. (5 warmup runs). 2 x CPU slowdown.	45.6 ±0.7 (1.04)	46.5 ±0.7 (1.06)	44.0 ±0.5 (1.00)	49.5 ±0.6 (1.13)
create many rows creating 10,000 rows. (5 warmup runs).	906.6 ±5.5 (1.01)	901.9 ±4.5 (1.00)	1,004.3 ±13.8 (1.11)	1,094.7 ±2.5 (1.21)
append rows to large table appending 1,000 to a table of 1,000 rows. (5 warmup runs).	107.7 ±0.8 (1.00)	108.2 ±3.3 (1.00)	125.1 ±2.4 (1.16)	131.8 ±2.6 (1.22)
clear rows clearing a table with 1,000 rows. (5 warmup runs). 4 x CPU slowdown.	34.3 ±0.8 (1.00)	39.0 ±0.7 (1.14)	41.3 ±0.8 (1.20)	42.9 ±1.2 (1.25)
weighted geometric mean of all factors in the table	1	1	0.9	0.85

Насколько репрезентативен бенчмарк?



Насколько репрезентативен бенчмарк?

0.0

Абсолютно никакого эффекта

Насколько репрезентативен бенчмарк?

0.0

Абсолютно никакого эффекта

Насколько репрезентативен бенчмарк?

Массивы в React - это списки

Насколько репрезентативен бенчмарк?

Массивы в React - это списки

<For /> в SolidJS и #each в Svelte, а не компиляция

Насколько репрезентативен бенчмарк?

Массивы в React - это списки

<For /> в SolidJS и #each в Svelte, а не компиляция

СИГНАЛЫ ~ React

ВСЁ!

性能

Performance

Longer is better!

* The results are based on js-framework-benchmark, and tested on a MacBook Pro M1 Max.

Геометрическое среднее

Если посмотреть на результаты, то видно, что в контрольных показателях факторы распределены неравномерно. Факторы, связанные с созданием строки, расположены ближе друг к другу, чем факторы, связанные с выбором строки, и так далее. Если мы просто возьмем геометрическое среднее, то подчеркнем влияние тех контрольных показателей, которые имеют большой разброс. (К сожалению, это даже самые слабые контрольные показатели с точки зрения дисперсии и стабильности...) Поэтому кажется целесообразным использовать взвешенное геометрическое среднее (https://en.wikipedia.org/wiki/Weighted_geometric_mean).

Среднее арифметическое ряда – это такое число, на которое можно заменить каждый элемент ряда так, чтобы их сумма не изменилась.

Среднее геометрическое – это такое число, на которое можно заменить все элементы ряда, чтобы их произведение не поменялось.

https://maximumtest.ru/uchebnik/9-klass/matematika/osnovy-matematicheskoy-statistiki

Name Duration for...	solid-v1.9.3	svelte-v5.42.1	react-hooks-v19.2.0	react-mobX-v19.0.0 + 6.13.5
Implementation notes				
Implementation link	code	code	code	code
create rows creating 1,000 rows. (5 warmup runs).	90.2 ± 1.1 (1.00)	92.8 ± 1.0 (1.03)	109.7 ± 1.0 (1.22)	118.6 ± 1.1 (1.31)
replace all rows updating all 1,000 rows. (5 warmup runs).	97.9 ± 1.2 (1.00)	99.4 ± 0.9 (1.02)	121.2 ± 1.0 (1.24)	130.7 ± 2.0 (1.34)
partial update updating every 10th row for 1,000 row. (3 warmup runs). 4 x CPU slowdown.	55.6 ± 0.4 (1.00)	56.4 ± 0.9 (1.01)	68.6 ± 0.7 (1.23)	72.5 ± 0.9 (1.30)
select row highlighting a selected row. (5 warmup runs). 4 x CPU slowdown.	14.5 ± 0.7 (1.00)	17.9 ± 1.0 (1.23)	18.2 ± 0.4 (1.26)	20.1 ± 0.8 (1.39)
swap rows swap 2 rows for table with 1,000 rows. (5 warmup runs). 4 x CPU slowdown.	65.4 ± 0.9 (1.01)	64.8 ± 1.4 (1.00)	370.0 ± 5.0 (5.71)	353.0 ± 3.3 (5.45)
remove row removing one row. (5 warmup runs). 2 x CPU slowdown.	47.9 ± 1.1 (1.01)	47.3 ± 1.1 (1.00)	53.6 ± 0.8 (1.13)	54.4 ± 0.7 (1.15)
create many rows creating 10,000 rows. (5 warmup runs).	937.4 ± 5.2 (1.00)	933.3 ± 8.4 (1.00)	1,314.0 ± 9.0 (1.41)	1,403.7 ± 11.4 (1.50)
append rows to large table appending 1,000 to a table of 1,000 rows. (5 warmup runs).	110.6 ± 1.2 (1.00)	110.8 ± 1.4 (1.00)	123.4 ± 0.9 (1.12)	132.6 ± 1.1 (1.20)
clear rows clearing a table with 1,000 rows. (5 warmup runs). 4 x CPU slowdown.	39.9 ± 0.9 (1.14)	35.0 ± 0.7 (1.00)	60.9 ± 0.8 (1.74)	65.0 ± 0.7 (1.86)
weighted geometric mean of all factors in the table	1.02	1.02	1.33	1.41

Геометрическое среднее

	A	B	C
Add	1	1	3
Update	1	4	3
Remove	7	4	3
Total	1.9	2.5	3
	$\sqrt[3]{(1\times1\times7)}$	$\sqrt[3]{(1\times4\times4)}$	$\sqrt[3]{(3\times3\times3)}$

Геометрическое среднее

	A	B	C
Add	1	1	3
Update	1	4	3
Remove	7	4	3
Total	1.9	2.5	3
	$\sqrt[3]{(1\times1\times7)}$	$\sqrt[3]{(1\times4\times4)}$	$\sqrt[3]{(3\times3\times3)}$

Геометрическое среднее

	A	B	C
Add	1	1	3
Update	1	4	3
Remove	7	4	3
Total	1.9	2.5	3

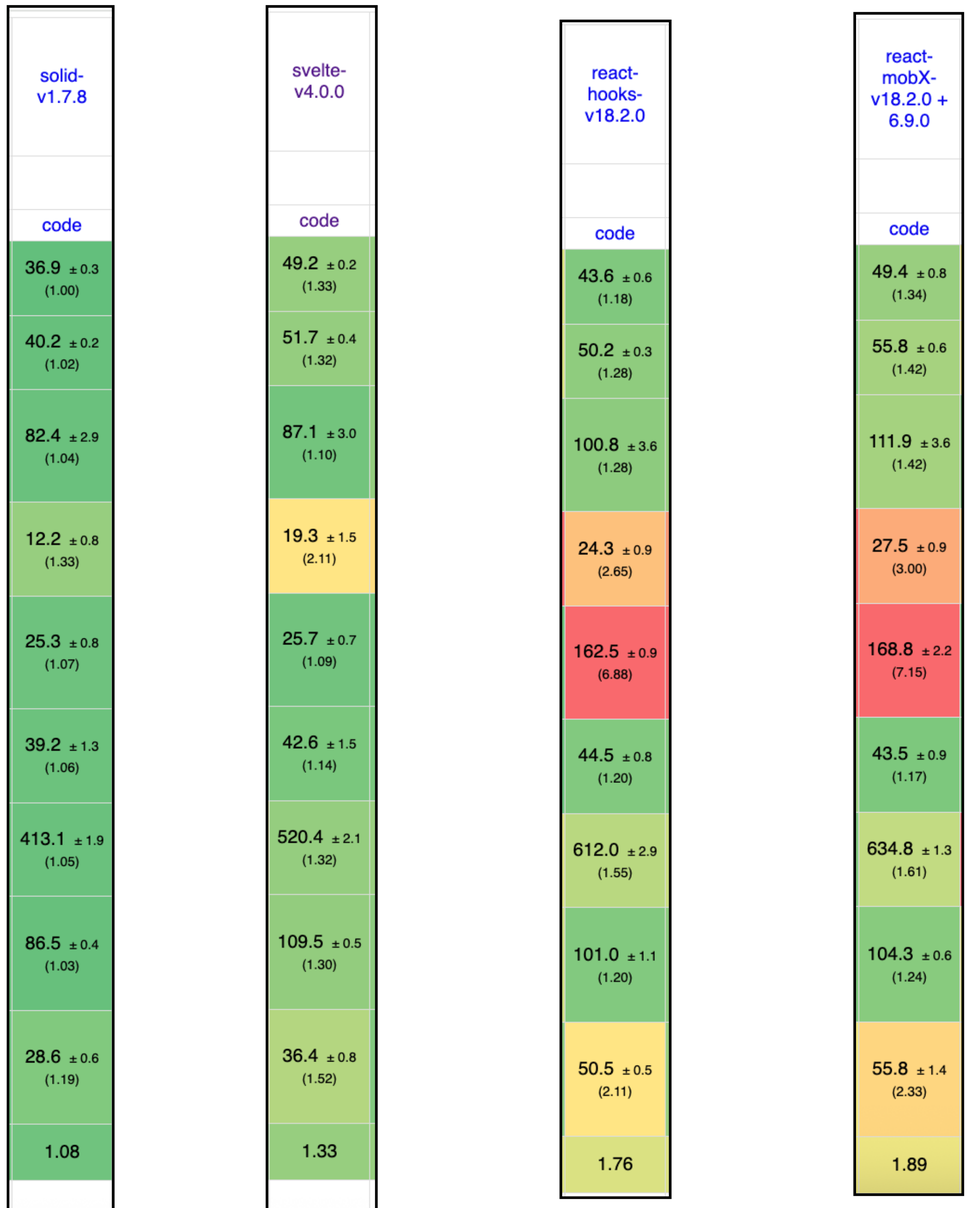
Геометрическое среднее

3

	A	B	C
Add	1	1	3
Update	1	4	3
Remove	7	4	3
Total	1.9	2.5	3

9 9 9

Chrome117


solid-v1.7.8
code
36.9 ± 0.3 (1.00)
40.2 ± 0.2 (1.02)
82.4 ± 2.9 (1.04)
12.2 ± 0.8 (1.33)
25.3 ± 0.8 (1.07)
39.2 ± 1.3 (1.06)
413.1 ± 1.9 (1.05)
86.5 ± 0.4 (1.03)
28.6 ± 0.6 (1.19)
1.08
svelte-v4.0.0
code
49.2 ± 0.2 (1.33)
51.7 ± 0.4 (1.32)
87.1 ± 3.0 (1.10)
19.3 ± 1.5 (2.11)
25.7 ± 0.7 (1.09)
42.6 ± 1.5 (1.14)
520.4 ± 2.1 (1.32)
109.5 ± 0.5 (1.30)
36.4 ± 0.8 (1.52)
1.33
react-hooks-v18.2.0
code
43.6 ± 0.6 (1.18)
50.2 ± 0.3 (1.28)
100.8 ± 3.6 (1.28)
24.3 ± 0.9 (2.65)
162.5 ± 0.9 (6.88)
44.5 ± 0.8 (1.20)
612.0 ± 2.9 (1.55)
101.0 ± 1.1 (1.20)
50.5 ± 0.5 (2.11)
1.76
react-mobX-v18.2.0 + 6.9.0
code
49.4 ± 0.8 (1.34)
55.8 ± 0.6 (1.42)
111.9 ± 3.6 (1.42)
27.5 ± 0.9 (3.00)
168.8 ± 2.2 (7.15)
43.5 ± 0.9 (1.17)
634.8 ± 1.3 (1.61)
104.3 ± 0.6 (1.24)
55.8 ± 1.4 (2.33)
1.89

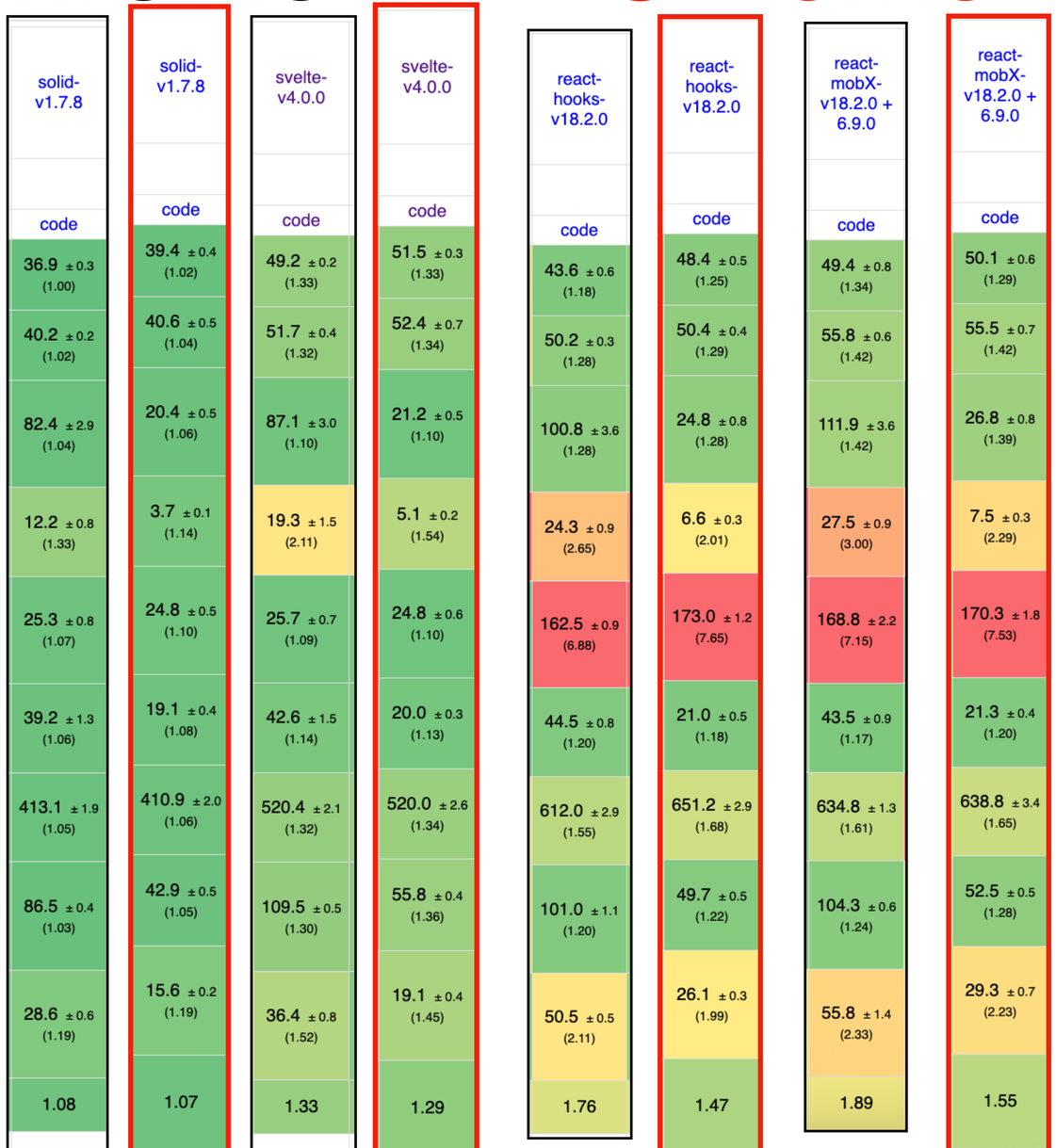
Это текущие значения веса:

бенчмарк	самый быстрый	90-й процентиль	90%-ный фактор	*масса*	самый медленный	фактор для самого медленного
01_run1k	38.72	60.24	1.56	0,64	126.51	3.27
02_replace1k	39.15	69.82	1.78	0,56	192.16	4.91
03_update10th1k_x16	19.34	34.26	1.77	0,56	168.47	8.71
04_select1k	3.29	17.06	5.19	0,19	156.59	47.67
05_swap1k	22.61	171.26	7.58	0,13	328.72	14.54
06_remove-one-1k	17.72	33.58	1.89	0,53	163.77	9.24
07_create10k	386.77	685.22	1.77	0,56	2415.89	6.25
08_create1k-after1k_x2	40.88	74.22	1.82	0,55	160.16	3.92
09_clear1k_x8	13.14	31.10	2.37	0,42	53.41	4.06

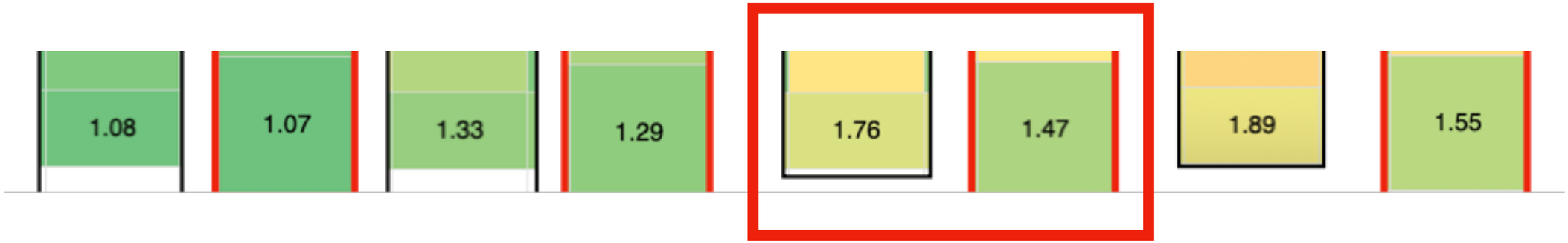
Эти весовые коэффициенты могут быть скорректированы в будущем.

Если хотите рассмотреть все подробнее, можете поработать с файлом Excel, содержащим данные о весах:
results.xlsx

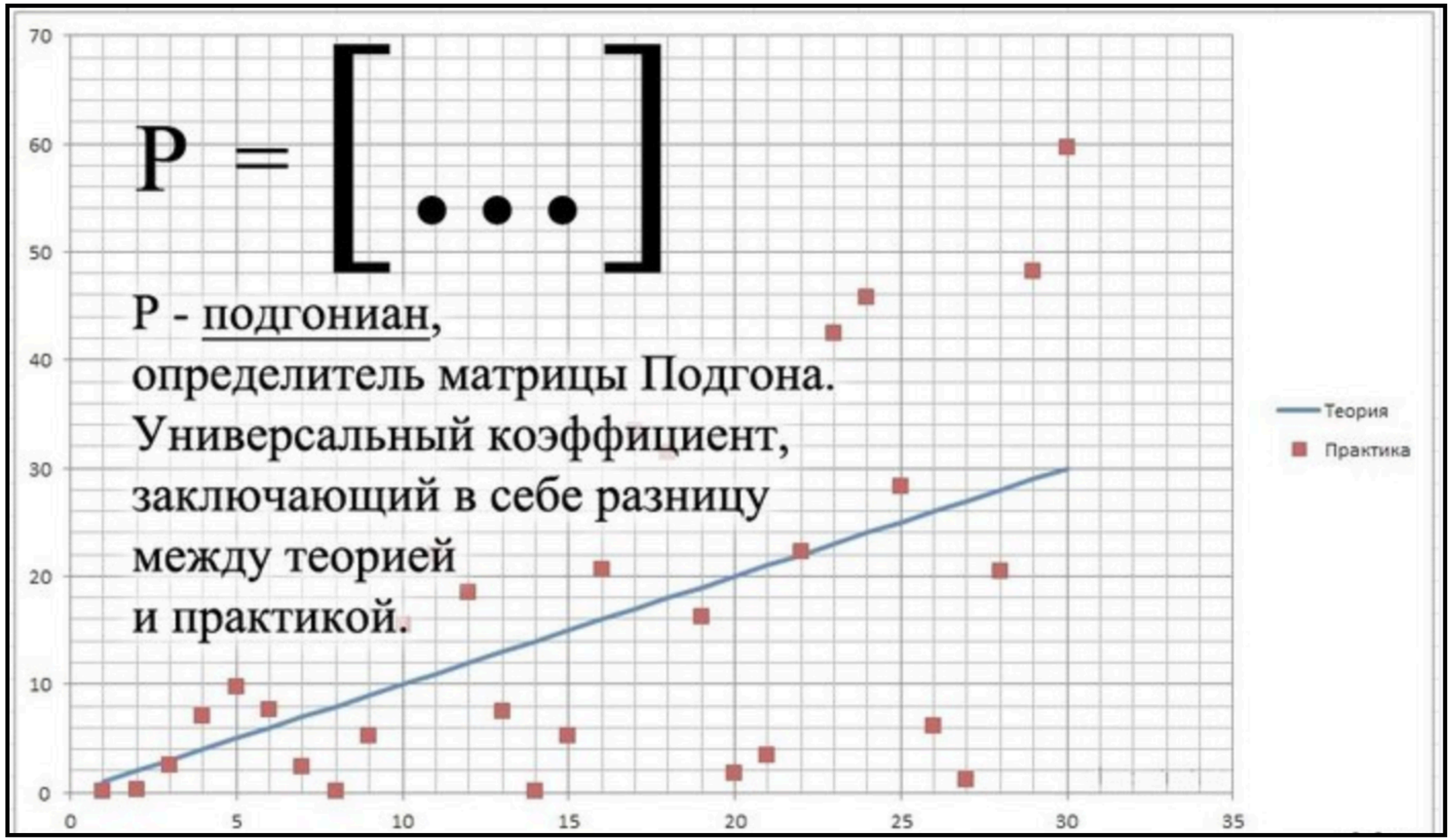
Chrome117 - Chrome 118



Chrome117 - Chrome 118



Геометрическое среднее



Геометрическое среднее

性能

Performance

Longer is better!

* The results are based on js-framework-benchmark, and tested on a MacBook Pro M1 Max.

Name Duration for...	svelte-v5.42.1	solid-v1.9.3	react-hooks-v19.2.0	react-mobX-v19.0.0 + 6.13.5
Implementation notes				
Implementation link	code	code	code	code
create rows creating 1,000 rows. (5 warmup runs).	90.3 ± 1.0 (1.02)	88.9 ± 0.8 (1.00)	103.5 ± 1.2 (1.16)	113.2 ± 0.8 (1.27)
replace all rows updating all 1,000 rows. (5 warmup runs).	97.3 ± 0.9 (1.01)	96.5 ± 0.9 (1.00)	111.9 ± 1.3 (1.16)	121.8 ± 1.3 (1.26)
partial update updating every 10th row for 1,000 row. (3 warmup runs). 4 x CPU slowdown.	52.0 ± 1.2 (1.01)	51.7 ± 0.4 (1.00)	61.2 ± 1.5 (1.18)	66.1 ± 1.1 (1.28)
select row highlighting a selected row. (5 warmup runs). 4 x CPU slowdown.	18.8 ± 0.9 (1.03)	18.4 ± 0.6 (1.01)	18.2 ± 0.8 (1.00)	21.2 ± 0.8 (1.16)
swap rows swap 2 rows for table with 1,000 rows. (5 warmup runs). 4 x CPU slowdown.	58.7 ± 0.8 (1.00)	59.9 ± 0.8 (1.02)	62.4 ± 1.9 (1.06)	63.3 ± 0.7 (1.08)
remove row removing one row. (5 warmup runs). 2 x CPU slowdown.	45.6 ± 0.7 (1.04)	46.5 ± 0.7 (1.06)	44.0 ± 0.5 (1.00)	49.5 ± 0.6 (1.13)
create many rows creating 10,000 rows. (5 warmup runs).	906.6 ± 5.5 (1.01)	901.9 ± 4.5 (1.00)	1,004.3 ± 13.8 (1.11)	1,094.7 ± 2.5 (1.21)
append rows to large table appending 1,000 to a table of 1,000 rows. (5 warmup runs).	107.7 ± 0.8 (1.00)	108.2 ± 3.3 (1.00)	125.1 ± 2.4 (1.16)	131.8 ± 2.6 (1.22)
clear rows clearing a table with 1,000 rows. (5 warmup runs). 4 x CPU slowdown.	34.3 ± 0.8 (1.00)	39.0 ± 0.7 (1.14)	41.3 ± 0.8 (1.20)	42.9 ± 1.2 (1.25)
weighted geometric mean of all factors in the table	1.01	1.02	1.13	1.22

Name Duration for...	svelte-v5.42.1	solid-v1.9.3	react-hooks-v19.2.0	react-mobX-v19.0.0 + 6.13.5
Implementation notes				
Implementation link	code	code	code	code
create rows creating 1,000 rows. (5 warmup runs).	1,079.8 ± 0.7 (1.00)	1,081.1 ± 0.8 (1.00)	1,099.2 ± 0.9 (1.02)	1,113.3 ± 1.8 (1.03)
replace all rows updating all 1,000 rows. (5 warmup runs).	2,077.7 ± 0.8 (1.00)	2,085.7 ± 0.7 (1.00)	2,104.3 ± 0.8 (1.01)	2,109.8 ± 1.2 (1.02)
partial update updating every 10th row for 1,000 row. (3 warmup runs). 4 x CPU slowdown.	145.0 ± 1.0 (1.01)	143.2 ± 0.8 (1.00)	157.2 ± 1.4 (1.10)	158.5 ± 1.3 (1.11)
select row highlighting a selected row. (5 warmup runs). 4 x CPU slowdown.	17.5 ± 1.4 (1.12)	16.9 ± 1.1 (1.08)	15.6 ± 0.7 (1.00)	18.9 ± 2.4 (1.21)
swap rows swap 2 rows for table with 1,000 rows. (5 warmup runs). 4 x CPU slowdown.	54.2 ± 5.6 (1.00)	58.3 ± 2.1 (1.08)	54.1 ± 2.1 (1.00)	63.1 ± 1.3 (1.17)
remove row removing one row. (5 warmup runs). 2 x CPU slowdown.	41.0 ± 1.2 (1.00)	45.5 ± 0.7 (1.11)	48.3 ± 0.9 (1.18)	49.8 ± 1.6 (1.21)
create many rows creating 10,000 rows. (5 warmup runs).	10,812.9 ± 10.9 (1.00)	10,817.5 ± 3.3 (1.00)	11,024.0 ± 5.9 (1.02)	11,106.5 ± 7.0 (1.03)
append rows to large table appending 1,000 to a table of 1,000 rows. (5 warmup runs).	1,090.7 ± 0.8 (1.00)	1,092.9 ± 1.3 (1.00)	1,115.5 ± 0.8 (1.02)	1,135.4 ± 1.5 (1.04)
clear rows clearing a table with 1,000 rows. (5 warmup runs). 4 x CPU slowdown.	1,032.9 ± 7.4 (1.02)	1,030.6 ± 0.7 (1.02)	1,010.7 ± 1.1 (1.00)	1,018.6 ± 1.8 (1.01)
weighted geometric mean of all factors in the table	1.01	1.02	1.04	1.07

CPU slowdown.				
swap rows swap 2 rows for table with 1,000 rows. (5 warmup runs). 4 x CPU slowdown.	58.7 ± 0.8 (1.00)	59.9 ± 0.8 (1.02)	62.4 ± 1.9 (1.06)	63.3 ± 0.7 (1.08)
remove row removing one row. (5 warmup runs). 2 x CPU slowdown.	45.6 ± 0.7 (1.04)	46.5 ± 0.7 (1.06)	44.0 ± 0.5 (1.00)	49.5 ± 0.6 (1.13)
create many rows creating 10,000 rows. (5 warmup runs).	906.6 ± 5.5 (1.01)	901.9 ± 4.5 (1.00)	1,004.3 ± 13.8 (1.11)	1,094.7 ± 2.5 (1.21)
append rows to large table appending 1,000 to a table of 1,000 rows. (5 warmup runs).	107.7 ± 0.8 (1.00)	108.2 ± 3.3 (1.00)	125.1 ± 2.4 (1.16)	131.8 ± 2.6 (1.22)
clear rows clearing a table with 1,000 rows. (5 warmup runs). 4 x CPU slowdown.	34.3 ± 0.8 (1.00)	39.0 ± 0.7 (1.14)	41.3 ± 0.8 (1.20)	42.9 ± 1.2 (1.25)
weighted geometric mean of all factors in the table	1.01	1.02	1.13	1.22

swap rows swap 2 rows for table with 1,000 rows. (5 warmup runs). 4 x CPU slowdown.	54.2 ± 5.6 (1.00)	58.3 ± 2.1 (1.08)	54.1 ± 2.1 (1.00)	63.1 ± 1.3 (1.17)
remove row removing one row. (5 warmup runs). 2 x CPU slowdown.	41.0 ± 1.2 (1.00)	45.5 ± 0.7 (1.11)	48.3 ± 0.9 (1.18)	49.8 ± 1.6 (1.21)
create many rows creating 10,000 rows. (5 warmup runs).	10,812.9 ± 10.9 (1.00)	10,817.5 ± 3.3 (1.00)	11,024.0 ± 5.9 (1.02)	11,106.5 ± 7.0 (1.03)
append rows to large table appending 1,000 to a table of 1,000 rows. (5 warmup runs).	1,090.7 ± 0.8 (1.00)	1,092.9 ± 1.3 (1.00)	1,115.5 ± 0.8 (1.02)	1,135.4 ± 1.5 (1.04)
clear rows clearing a table with 1,000 rows. (5 warmup runs). 4 x CPU slowdown.	1,032.9 ± 7.4 (1.02)	1,030.6 ± 0.7 (1.02)	1,010.7 ± 1.1 (1.00)	1,018.6 ± 1.8 (1.01)
weighted geometric mean of all factors in the table	1.01	1.02	1.04	1.07