JitTeam™

Performance optimization in React

front.jit #1 (28.06.2022)

https://jit.team

Maciej Kankowski

- In Jit Team for 50 months 🤠
- Frontend (React/Next/Vue + 🎨)
- Mobile (🍏)
- Internal projects / mentoring 👨🏼‍🎓 👨🏼‍🎓
- Travel, drones, books, cars 🚀

Today's agenda

- Short introduction to optimization
- Is React fast?
- Solving performance issues by examples

Performance optimization

- Need for optimization
- **Measurement and tools**
- Choice of solution

Strategie optymalizacyjne

Runtime

- **Wyodrębnienie komponentu**
- **Wzorzec kompozycji**
- **Wirtualizacja**

- SSR
- WebWorkers
- „Lazy loading”
- Niemutowalność
- …

Buildtime

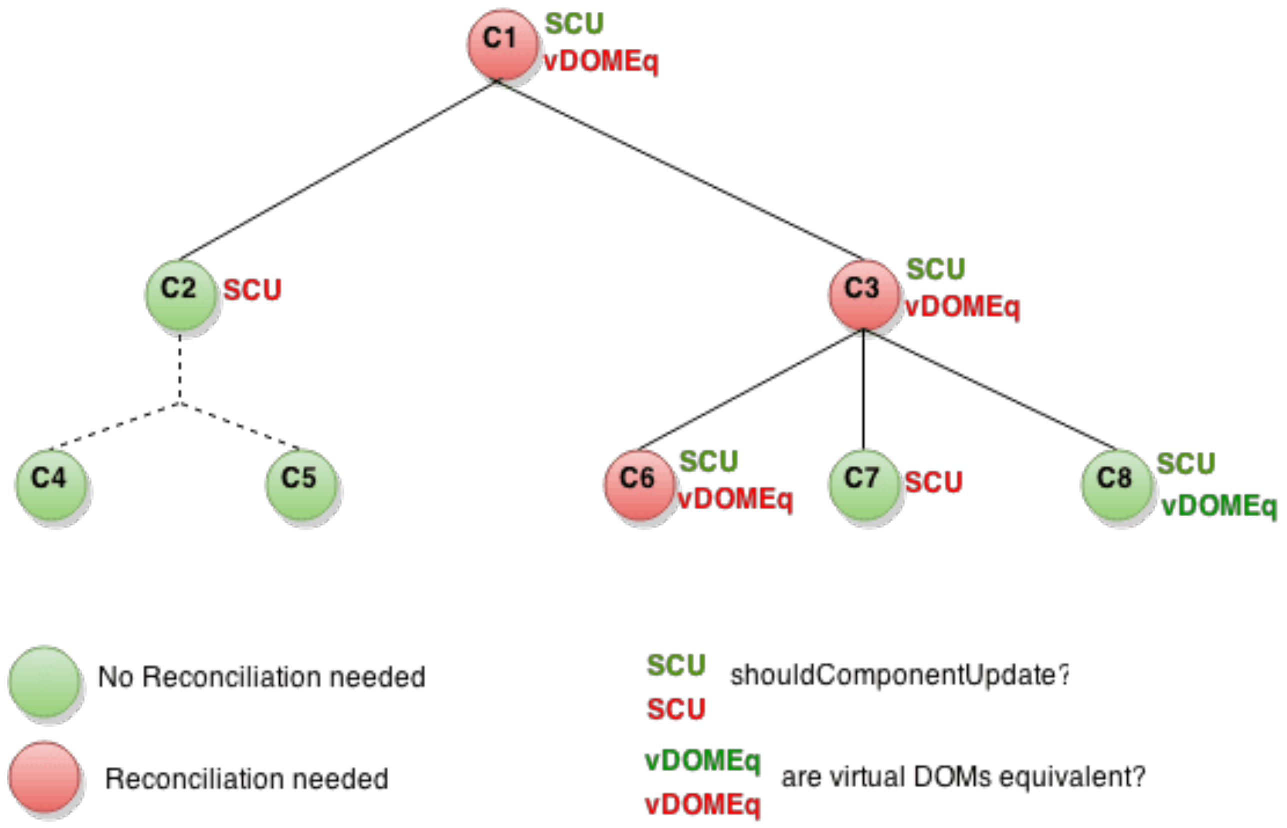
- Webpack, Vite, Rollup, ESBuild, Browserify
- wariant produkcyjny a deweloperski
- „analizery (np.. webpack-bundle-analyzer)
- „code splitting” (np. dla wielu punktów wejścia do aplikacji)
- „lazy loading” (doładowywanie ścieżek i komponentów)
- usunięcie zbędnego kodu („tree shaking”)

React is FAST! 🚀

React is FAST! 🚀
*not always

React

- Virtual DOM
- Key attribute
- Memoization



https://reactjs.org/docs/optimizing-performance.html

useMemo

- In the future, React may choose to "forget" some previously memoized values and recalculate them on next render, e.g. to free memory for offscreen components. Write your code so that it still works without useMemo — and then add it to optimize performance.
- We should also not use useMemo when the function returns a primitive value, such as a boolean or a string. Because primitive values are passed by value, not by reference, it means that they always remain the same, even if the component is re-rendered.
- If you're performing an operation that's not expensive (think Big O notation), then you don't need to memoize the return value. The cost of using useMemo may outweigh the cost of reevaluating the function.

useCallback

❌ “Every callback function should be memoized to prevent useless re-rendering of child components that use the callback function”

✅ Big list of items: Memoization of the row cell in memoized parent component to prevent re-renders (and avoid break of the parent memorization)

console.warn('Talk is cheap, show me your code!')

Example #1: Unresponsive (slow) component

Performance issue	Potential solution
Slow or unnecessary reloading of a component that has not changed state	Trace the state flow, identify what affects the unnecessary render and fix the source of the problems

https://github.com/mackankowski/front.jit/tree/main/src/lectures/performance-optimization/samples/slow-component

Example #2: Slow-loading list/table

Performance issue	Potential solution
A lot of elements to be displayed at the same time (e.g. a multiline table with a form)	Virtualization („windowing”) *vs. paginacja vs. lazy-loading…*

https://github.com/mackankowski/front.jit/tree/main/src/lectures/performance-optimization/samples/virtualized-table

Podsumowanie

- Optimization – on demand
- Using tools for specific problem resolving
- One issue = many solutions

Performance optimizations are not free. They ALWAYS come with a cost but do NOT always come with a benefit to offset that cost.

Kent C. Dodds

WE WANT YOU
TO CONTRIBUTE
FRONT.JIT

Links

Articles:

- https://reactjs.org/docs/optimizing-performance.html
- https://reactjs.org/docs/hooks-reference.html#usememo
- https://dmitripavlutin.com/use-react-memo-wisely/
- https://dmitripavlutin.com/dont-overuse-react-usecallback/
- https://overreacted.io/before-you-memo/
- https://kentcdodds.com/blog/usememo-and-usecallback
- https://blog.logrocket.com/rethinking-hooks-memoization/
- https://medium.com/@paularmstrong/twitter-lite-and-high-performance-react-progressive-web-apps-at-scale-d28a00e780a3
- https://www.patterns.dev/posts/virtual-lists/
- https://blog.logrocket.com/guide-performance-optimization-webpack/
  https://brycedooley.com/debug-react-rerenders/

Tools:

- Chrome DevTools Profiler with "Why did this render" feature
  https://reactjs.org/docs/profiler.html
- https://github.com/welldone-software/why-did-you-render
- https://gist.github.com/mackankowski/53843a02399f4dbac5b972624c24dc6b
- https://github.com/bvaughn/react-window
- https://github.com/bvaughn/react-window-infinite-loader