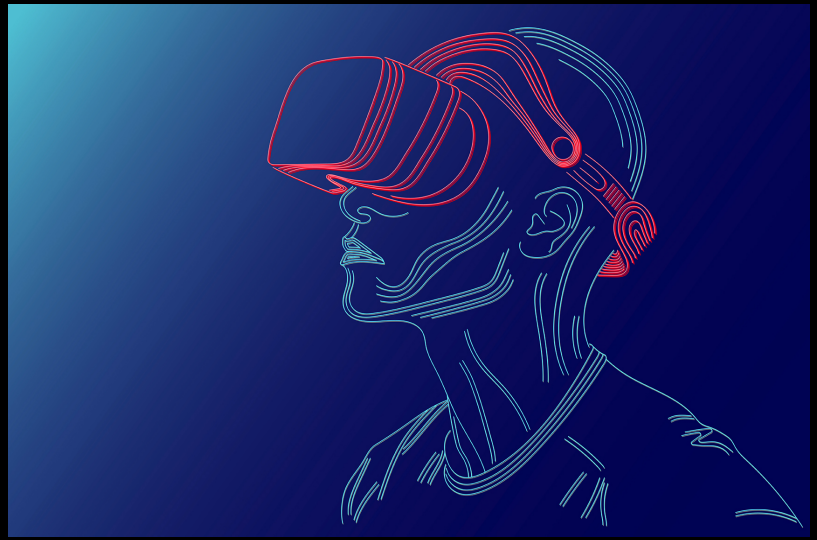
Mash-Up

Virtual Scroll

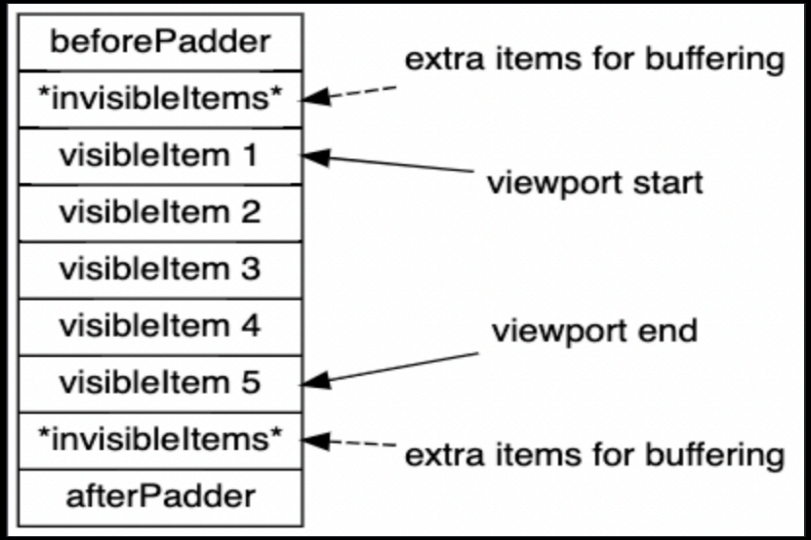
장원진

Virtual ?

What ?

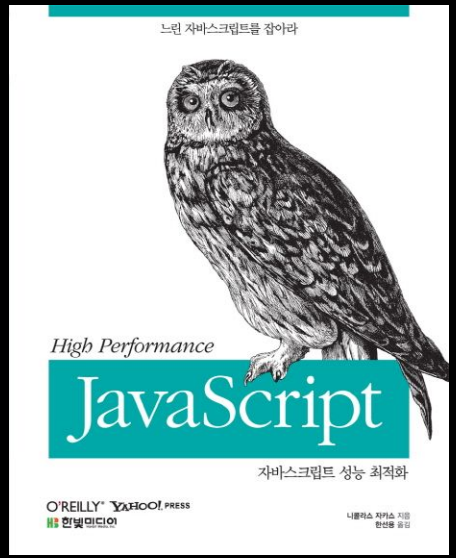


What ?


beforePadder
invisibleItems
visibleItem 1
visibleItem 2
visibleItem 3
visibleItem 4
visibleItem 5
invisibleItems
afterPadder
extra items for buffering
viewport start
viewport end
extra items for buffering

What's in it for me ?

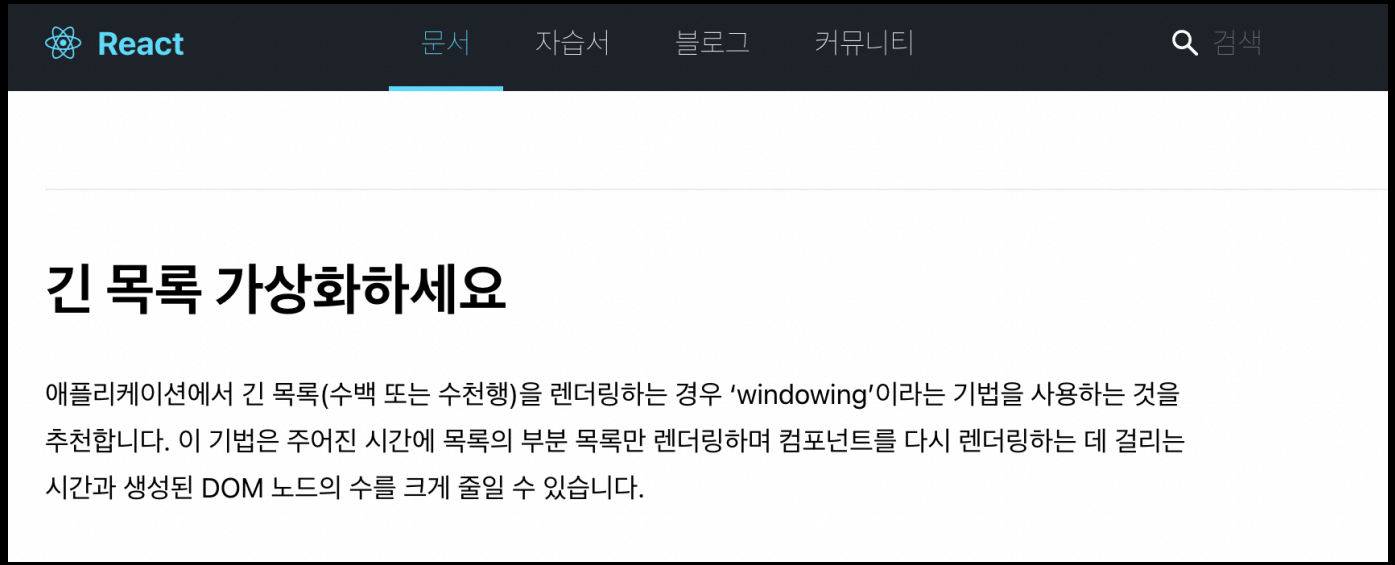
Why ?



“DOM은 태생부터 느립니다”

- 니콜라스 자카스

Why ?


React
문서
자습서
블로그
커뮤니티
검색
긴 목록 가상화하세요
애플리케이션에서 긴 목록(수백 또는 수천행)을 렌더링하는 경우 'windowing'이라는 기법을 사용하는 것을 추천합니다. 이 기법은 주어진 시간에 목록의 부분 목록만 렌더링하며 컴포넌트를 다시 렌더링하는 데 걸리는 시간과 생성된 DOM 노드의 수를 크게 줄일 수 있습니다.

DOM 수를 줄일 수 있다

그럼 언제 써 ?

대규모 데이터를 효율적으로 렌더링

구현해보자 !

react-virtualized

react-window

How ?

```tsx
// react-virtualized CellMeasure과 동일한 기능.
const getItemSize = (index: number) => {
  const title = DUMMY_DATA[index].title;
  const totalCount = title.length;
  const itemHeight = 20;
  const padding = 10;

  // 위의 로직과 관계없이 결론적으로 아이템 index에 직접 접근하여 컨텐츠 크기를 할당하기 위함
  return totalCount * itemHeight + padding * 2;
};

return (
  // FixedSizeList의 대체. react-virtualized의 AutoSizer
  <VariableSizeList
    height={height}
    itemCount={DUMMY_DATA.length}
    itemSize={getItemSize}
    width={500}
    onScroll={({scrollOffset}) => {
      setScroll(scrollOffset);
    }}
    initialScrollOffset={Number(sessionStorage.getItem('scroll')) ?? 0}
  >
    {DataCard}
  </VariableSizeList>
```

How ?

```
// react-virtualized CellMeasure과 동일한 기능.
const getItemSize = (index: number) => {
  const title = DUMMY_DATA[index].title;
  const totalCount = title.length;
  const itemHeight = 20;
  const padding = 10;

  // 위의 로직과 관계없이 결론적으로 아이템 index에 직접 접근하여 컨텐츠 크기를 할당하기 위함
  return totalCount * itemHeight + padding * 2;
};

return (
  // FixedSizeList의 대체. react-virtualized의 AutoSizer
  <VariableSizeList
    height={height}
    itemCount={DUMMY_DATA.length}
    itemSize={getItemSize}
    width={500}
    onScroll={({scrollOffset}) => {
      setScroll(scrollOffset);
    }}
    initialScrollOffset={Number(sessionStorage.getItem('scroll')) ?? 0}
  >
    {DataCard}
  </VariableSizeList>
```

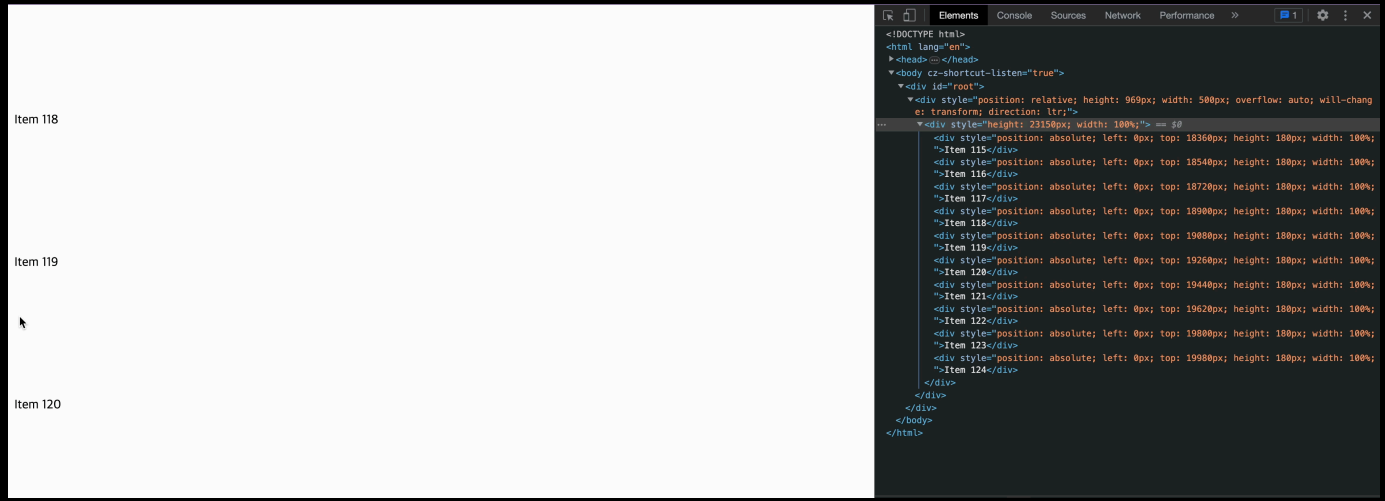
How ?

```
// react-virtualized CellMeasure과 동일한 기능.
const getItemSize = (index: number) => {
  const title = DUMMY_DATA[index].title;
  const totalCount = title.length;
  const itemHeight = 20;
  const padding = 10;

  // 위의 로직과 관계없이 결론적으로 아이템 index에 직접 접근하여 컨텐츠 크기를 할당하기 위함
  return totalCount * itemHeight + padding * 2;
};

return (
  // FixedSizeList의 대체. react-virtualized의 AutoSizer
  <VariableSizeList
    height={height}
    itemCount={DUMMY_DATA.length}
    itemSize={getItemSize}
    width={500}
    onScroll={({scrollOffset}) => {
      setScroll(scrollOffset);
    }}
    initialScrollOffset={Number(sessionStorage.getItem('scroll')) ?? 0}
  >
    {DataCard}
  </VariableSizeList>
```

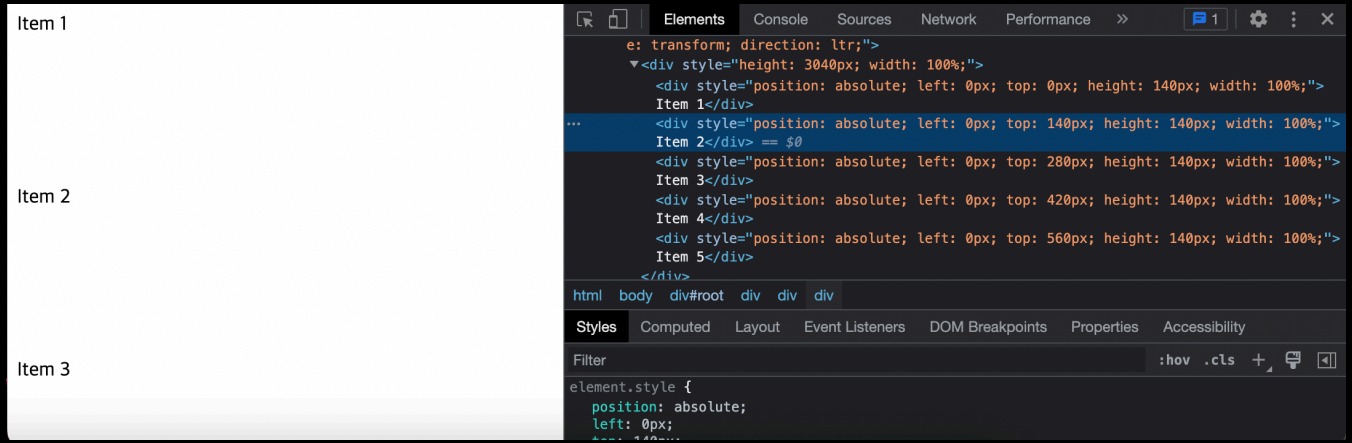
How ?


Item 118
Item 119
Item 120
Elements
Console
Sources
Network
Performance
<!DOCTYPE html>
<html lang="en">
<head> ⋯ </head>
<body cz-shortcut-listen="true">

 == $0
Item 115
Item 116
Item 117
Item 118
Item 119
Item 120
Item 121
Item 122
Item 123
Item 124

</body>
</html>

How ?


Item 1
Item 2
Item 3
Elements
Console
Sources
Network
Performance
e: transform; direction: ltr;">

Item 1

Item 2 == $0

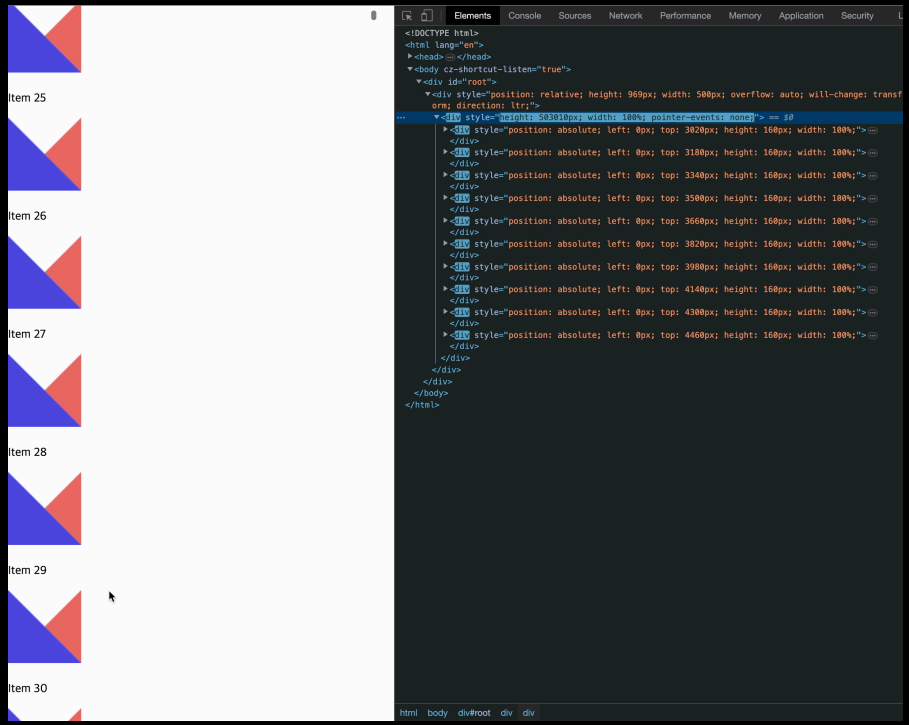
Item 3

Item 4

Item 5

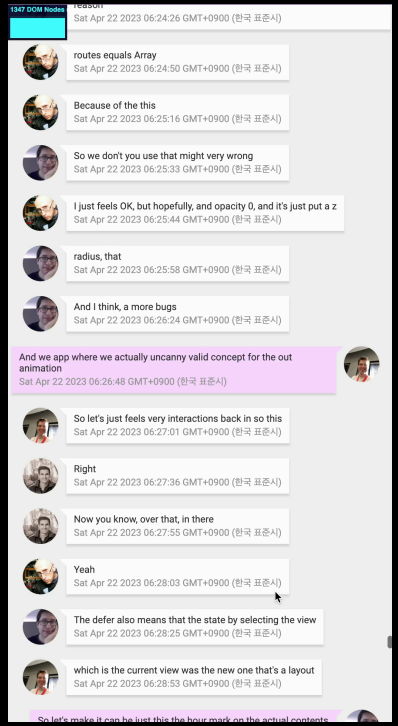
html body div#root div div div
Styles
Computed
Layout
Event Listeners
DOM Breakpoints
Properties
Accessibility
Filter
:hov .cls
element.style {
position: absolute;
left: 0px;

How ?


Item 25
Item 26
Item 27
Item 28
Item 29
Item 30
Elements
Console
Sources
Network
Performance
Memory
Application
Security
<!DOCTYPE html>
<html lang="en">
<head> … </head>
<body cz-shortcut-listen="true">

 == $0

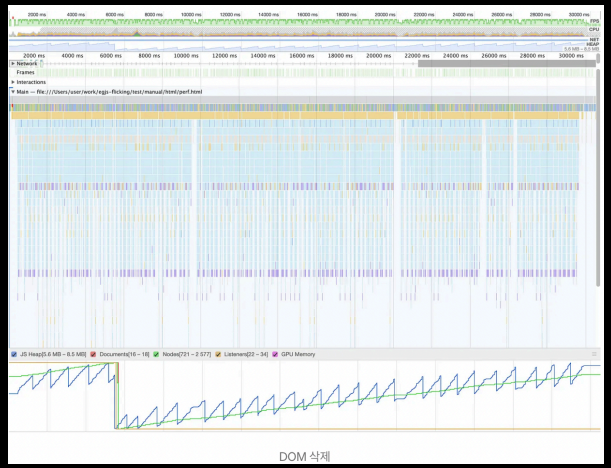
</body>
</html>
html
body
div#root
div
div


1347 DOM Nodes
Sat Apr 22 2023 06:24:26 GMT+0900 (한국 표준시)
routes equals Array
Sat Apr 22 2023 06:24:50 GMT+0900 (한국 표준시)
Because of the this
Sat Apr 22 2023 06:25:16 GMT+0900 (한국 표준시)
So we don't you use that might very wrong
Sat Apr 22 2023 06:25:33 GMT+0900 (한국 표준시)
I just feels OK, but hopefully, and opacity 0, and it's just put a z
Sat Apr 22 2023 06:25:44 GMT+0900 (한국 표준시)
radius, that
Sat Apr 22 2023 06:25:58 GMT+0900 (한국 표준시)
And I think, a more bugs
Sat Apr 22 2023 06:26:24 GMT+0900 (한국 표준시)
And we app where we actually uncanny valid concept for the out animation
Sat Apr 22 2023 06:26:48 GMT+0900 (한국 표준시)
So let's just feels very interactions back in so this
Sat Apr 22 2023 06:27:01 GMT+0900 (한국 표준시)
Right
Sat Apr 22 2023 06:27:36 GMT+0900 (한국 표준시)
Now you know, over that, in there
Sat Apr 22 2023 06:27:55 GMT+0900 (한국 표준시)
Yeah
Sat Apr 22 2023 06:28:03 GMT+0900 (한국 표준시)
The defer also means that the state by selecting the view
Sat Apr 22 2023 06:28:25 GMT+0900 (한국 표준시)
which is the current view was the new one that's a layout
Sat Apr 22 2023 06:28:53 GMT+0900 (한국 표준시)

react-window-infinite-loader

caching

아무런 수정도 가하지 않은 경우



DOM 삭제

- https://www.bucketplace.com/post/2020-09-10-%EC%98%A4%EB%8A%98%EC%9D%98%EC%A7%91-%EB%82%B4-%EB%AC%B4%ED%95%9C%EC%8A%A4%ED%81%AC%EB%A1%A4-%EA%B0%9C%EB%B0%9C%EA%B8%B0/
- https://developer.chrome.com/blog/infinite-scroller
- https://dev.to/akshaykumar6/building-high-performance-infinite-lists-in-react-31k7
- https://velog.io/@pandati0710/virtual-scroll%EC%9D%84-%EC%9D%B4%EC%9A%A9%ED%95%9C-%EC%84%B1%EB%8A%A5-%EC%B5%9C%EC%A0%81%ED%99%94
- https://crmrelease.tistory.com/108 (라이브러리x 구현)

감사합니다