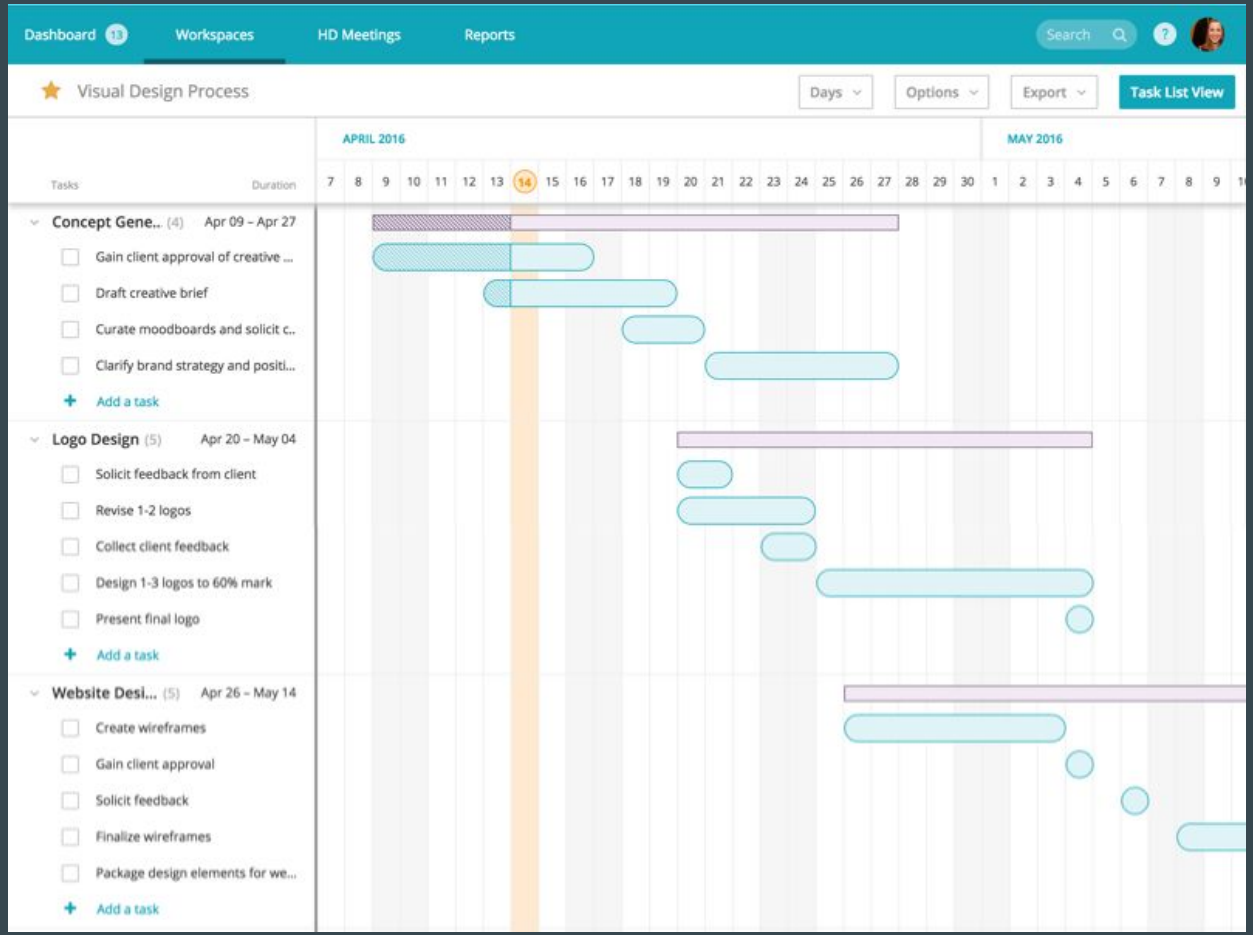
Intro to react virtualized

Redbooth gantt


Dashboard 13
Workspaces
HD Meetings
Reports
Search
Visual Design Process
Days
Options
Export
Task List View
APRIL 2016
MAY 2016
Tasks
Duration
7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 1 2 3 4 5 6 7 8 9
Concept Gene.. (4) Apr 09 – Apr 27
Gain client approval of creative ...
Draft creative brief
Curate moodboards and solicit c..
Clarify brand strategy and positi...
Add a task
Logo Design (5) Apr 20 – May 04
Solicit feedback from client
Revise 1-2 logos
Collect client feedback
Design 1-3 logos to 60% mark
Present final logo
Add a task
Website Desi... (5) Apr 26 – May 14
Create wireframes
Gain client approval
Solicit feedback
Finalize wireframes
Package design elements for we...
Add a task

Gantt prototype

Not MVP ready, yet

Make it to production

10K project

A project with just 10,000 tasks

Results

Slow initial rendering

Laggy scrolling

Really bad UX

✓ shouldComponentUpdate

✓ Redux

🤔 DOM

Problem

How to display large amount of data efficiently...

in a reasonable amount of time...

without freezing the ui...

without crashing the browser?

Virtual rendering

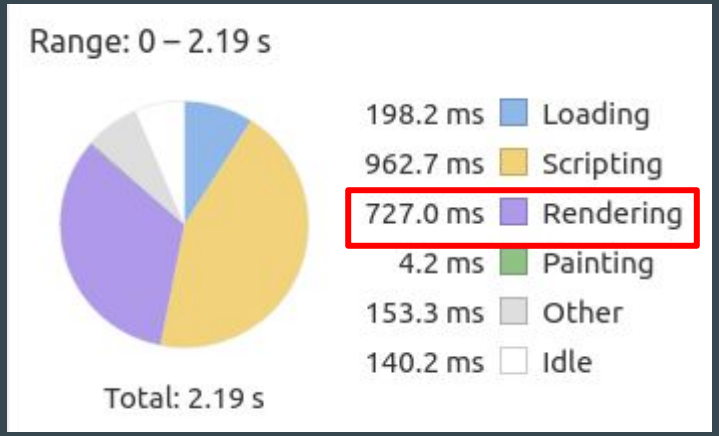

VIRTUAL WHAT?
imgflip.com

Virtual rendering

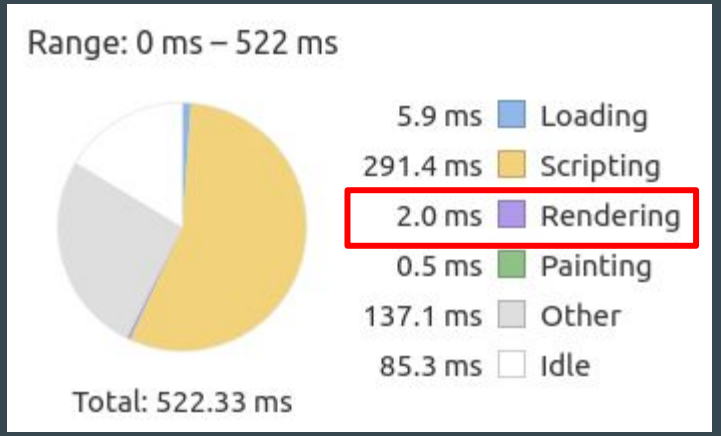
Render **only visible** stuff

Rendering 20K divs

Non virtualized



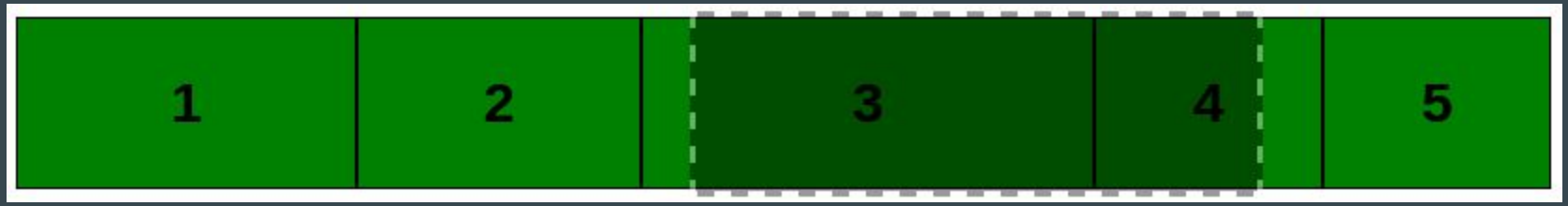
Virtualized



Virtual rendering

Windowing + Positioning

Windowing



Positioning

```
<div class="outer" style="width: 300px; height: 200px; overflow-y: scroll;">
  <div class="inner" style="position: relative; height: 400px;">
    <div style="position: absolute; top: 0; height: 50px; width: 100%;">
      <div class="item">Item 1</div>
    </div>
    <div style="position: absolute; top: 50px; height: 50px; width: 100%;">
      <div class="item">Item 2</div>
    </div>
    <div style="position: absolute; top: 100px; height: 50px; width: 100%;">
      <div class="item">Item 3</div>
    </div>
  </div>
</div>
```

Don’t reinvent the wheel,
It’s **already done**

react-virtualized

Actively maintained

Tabular and scattered layouts

Absolute positioning

Components

Grid

Great for tabular data

JIT measurement

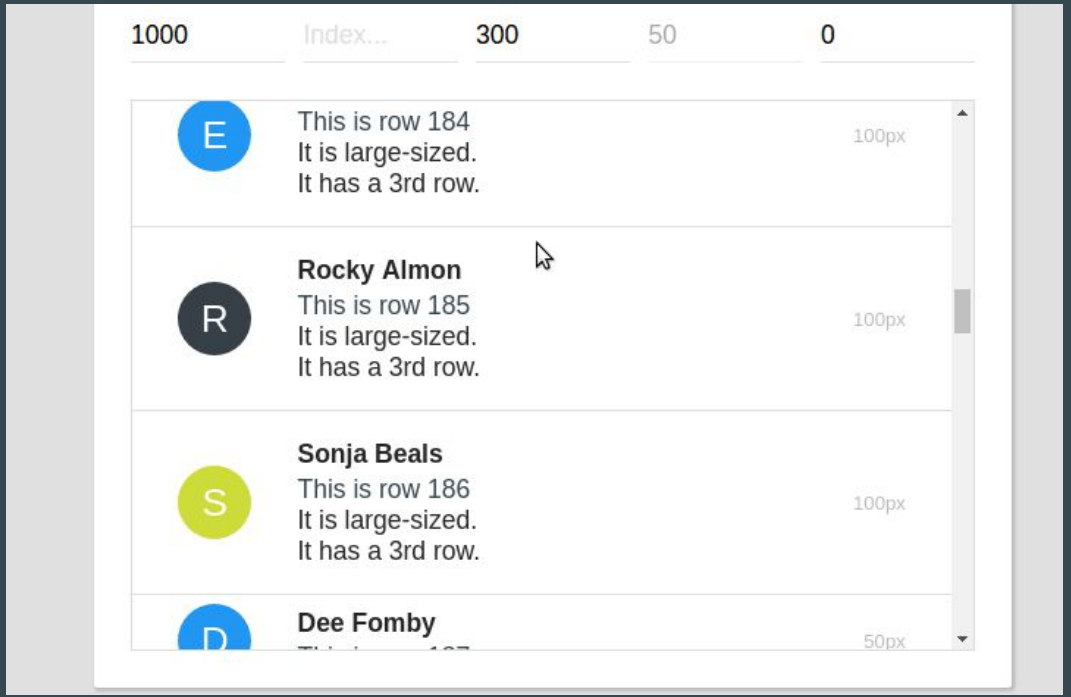
Dynamic row height

VirtualScroll

Good for simple lists

Infinite loading support

Extends Grid

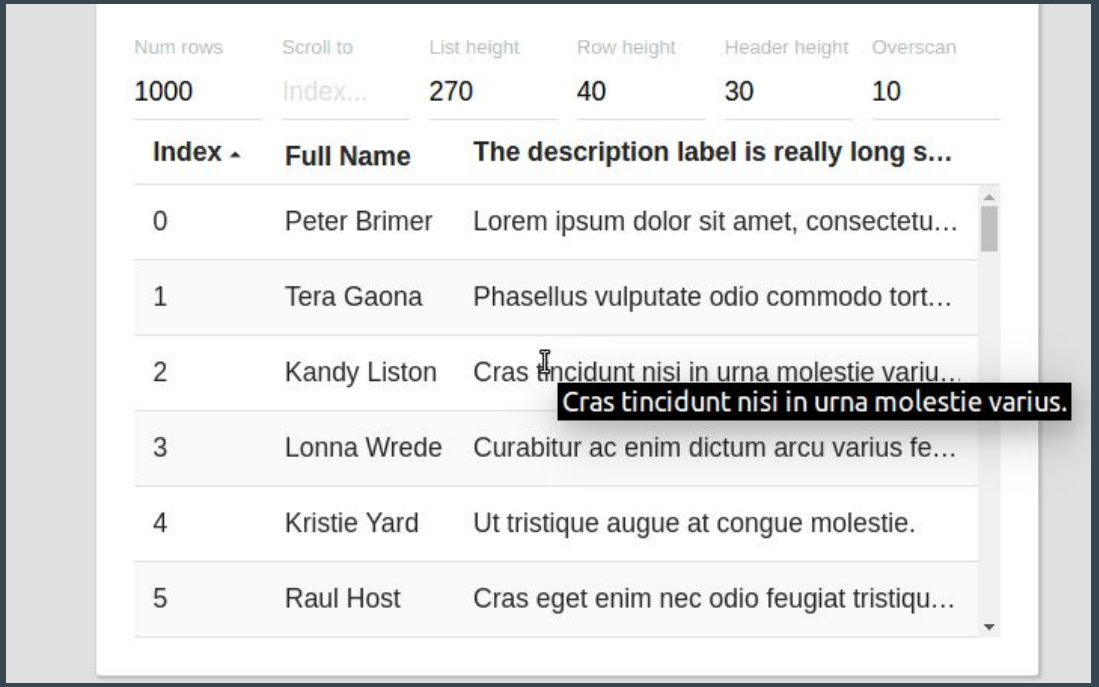


FlexTable

Table with built-in header

Sortable columns

Extends Grid

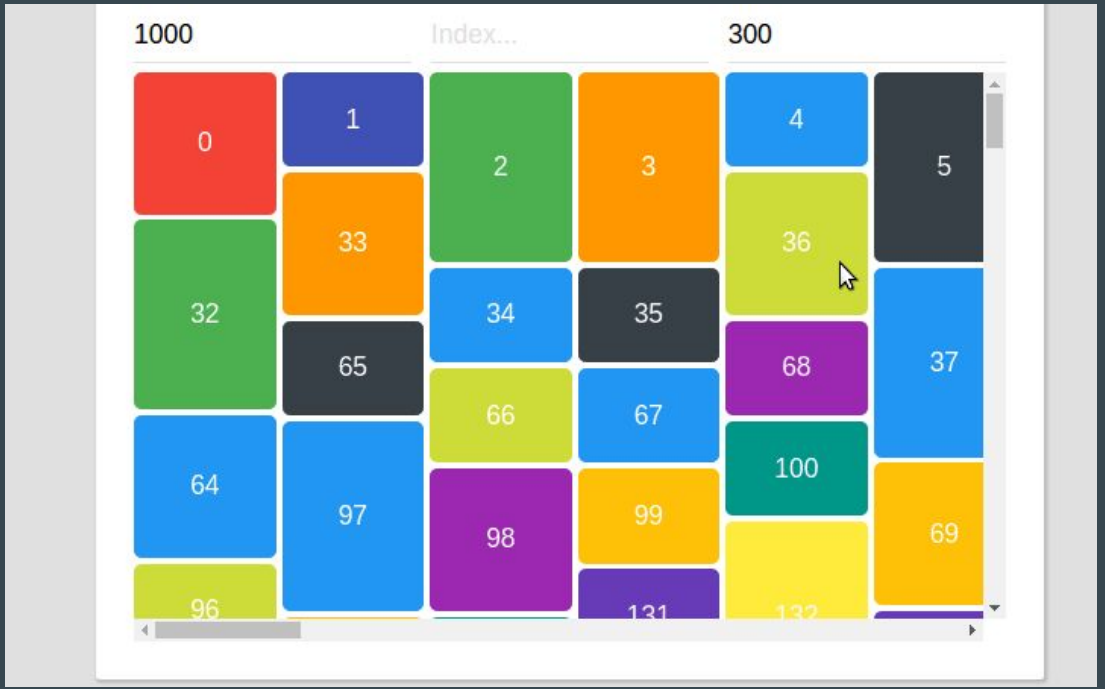


Collection

Scattered data

Flexible layout

Precomputed measurements



Utility

AutoSizer

ScrollSync

InfiniteLoader

CellMeasurer

Migrating to react virtualized

```
const items = [
  {label: “row 1”},
  {label: “row 2”},
  {label: “row 3”},
]
```

```
render() {
   <div>
      {this.props.items.map(this.renderRow)}
   </div>
}
```

```
renderRow(item) {
   <div className=”row”>
      {item.label}
   </div>
}
```

```
render() {
  <VirtualScroll
    width={300}
    height={100}
    rowHeight={50}
    rowCount={this.props.items.length}
    rowRenderer={this.renderRow}
  />
}
```

```
render() {
  <VirtualScroll
    width={300}
    height={100}
    rowHeight={50}
    rowCount={this.props.items.length}
    rowRenderer={this.renderRow}
  />
}
```

```
render() {
  <VirtualScroll
    width={300}
    height={100}
    rowHeight={50}
    rowCount={this.props.items.length}
    rowRenderer={this.renderRow}
  />
}
```

```
render() {
  <VirtualScroll
    width={300}
    height={100}
    rowHeight={50}
    rowCount={this.props.items.length}
    rowRenderer={this.renderRow}
  />
}
```

```
render() {
  <VirtualScroll
    width={300}
    height={100}
    rowHeight={50}
    rowCount={this.props.items.length}
    rowRenderer={this.renderRow}
  />
}
```

```
renderRow({ index }) {
  const item = this.props.items[index];

  <div className="row">
    {item.label}
  </div>
}
```

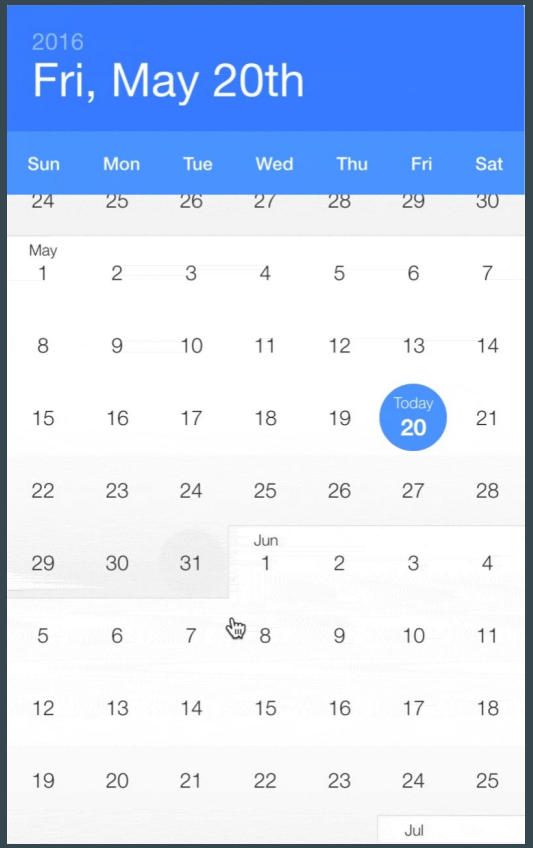
```
renderRow({ index }) {
  const item = this.props.items[index];

  <div className="row">
    {item.label}
  </div>
}
```

Real world examples

Infinite calendar

VirtualScroll

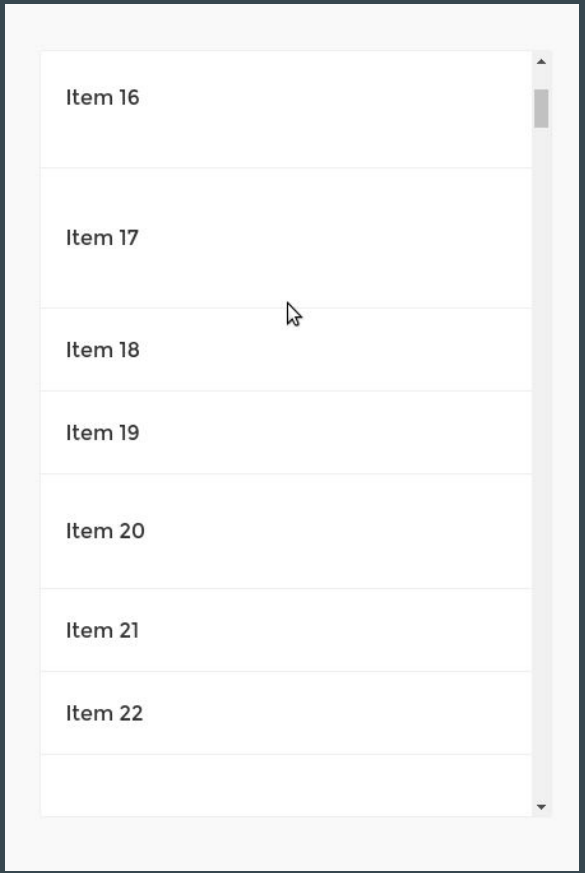
- Infinite scroll
- Smooth animations



Sortable list

VirtualScroll

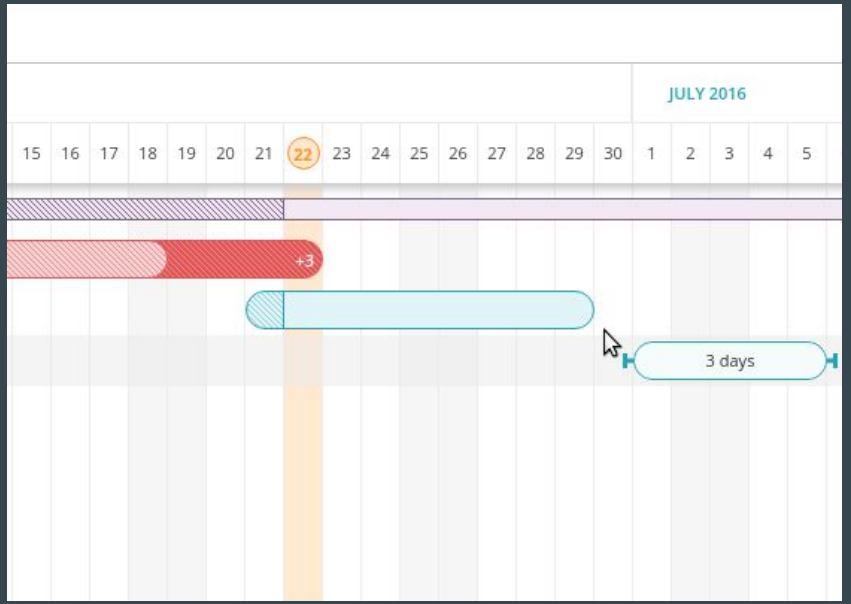
- Touch support
- Drag&Drop



Redbooth gantt

Grid + Collection

- Drag&Drop
- Synced grids



Usage considerations

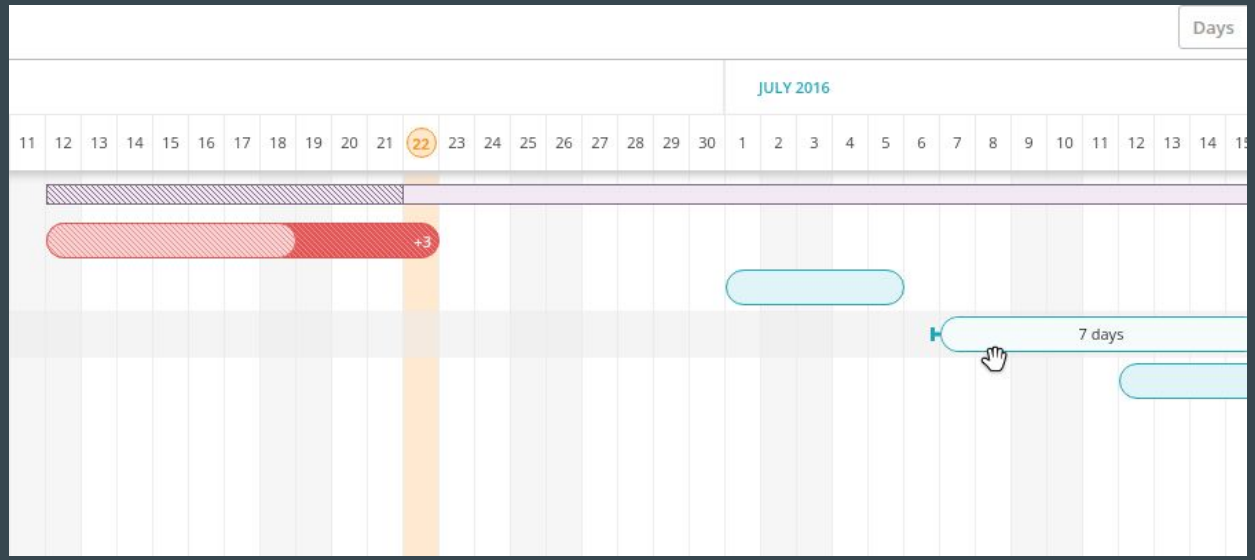
Data changes

“VirtualScroll has no way of knowing when its underlying list data has changed since it only receives a rowHeight property”

Use VirtualScroll.recomputeRowHeights

Single cell update

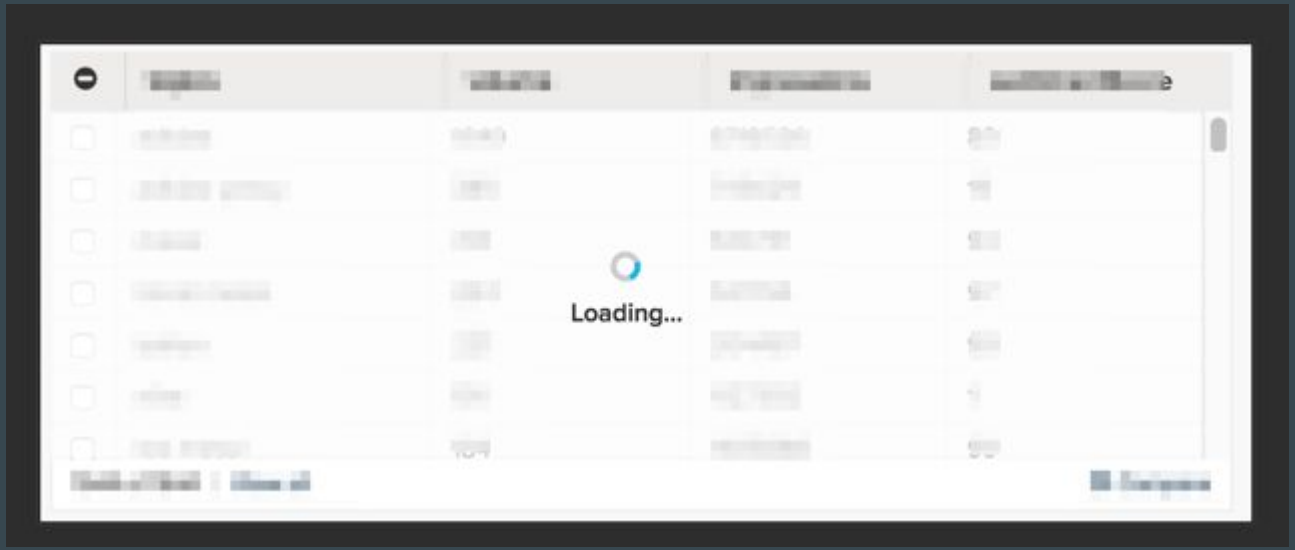
State is not preserved



Connect to a store (Redux, Flux, MobX...)

Custom cell layout

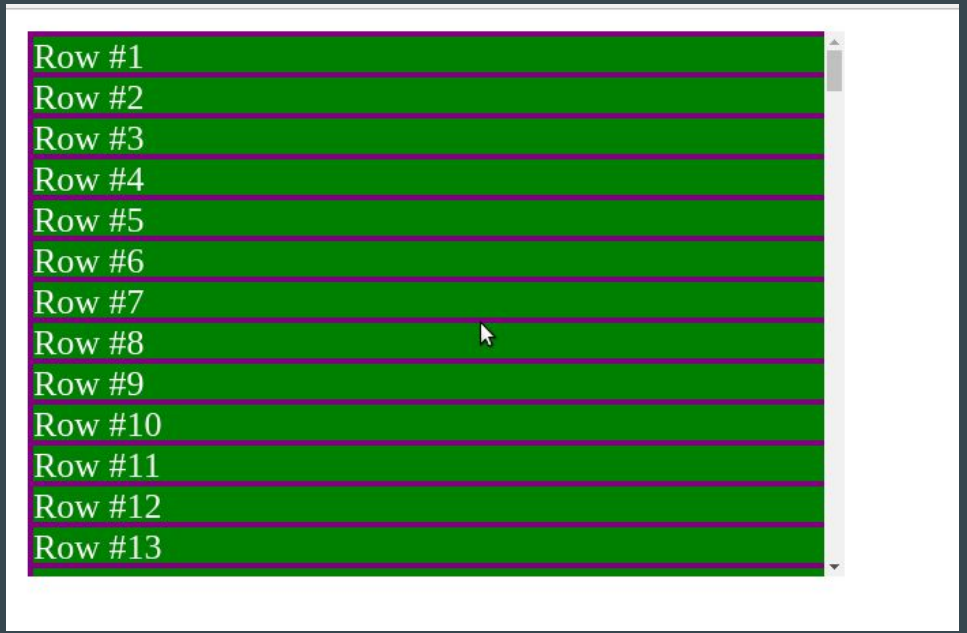
Useful for cell overlays



Use cellRangeRenderer

Scroll flickering

Small cells



Use overscan option

Getting started

- ❑ Check the demo website
- ❑ Check the examples code
- ❑ Chat on Gitter

Q&A

Resources

http://codepen.io/edulan/pen/pbENpq

http://www.lab4games.net/zz85/blog/2012/06/23/virtual-rendering-1000000-items-efficiently/

https://github.com/clauderic/react-sortable-hoc

https://github.com/clauderic/react-infinite-calendar