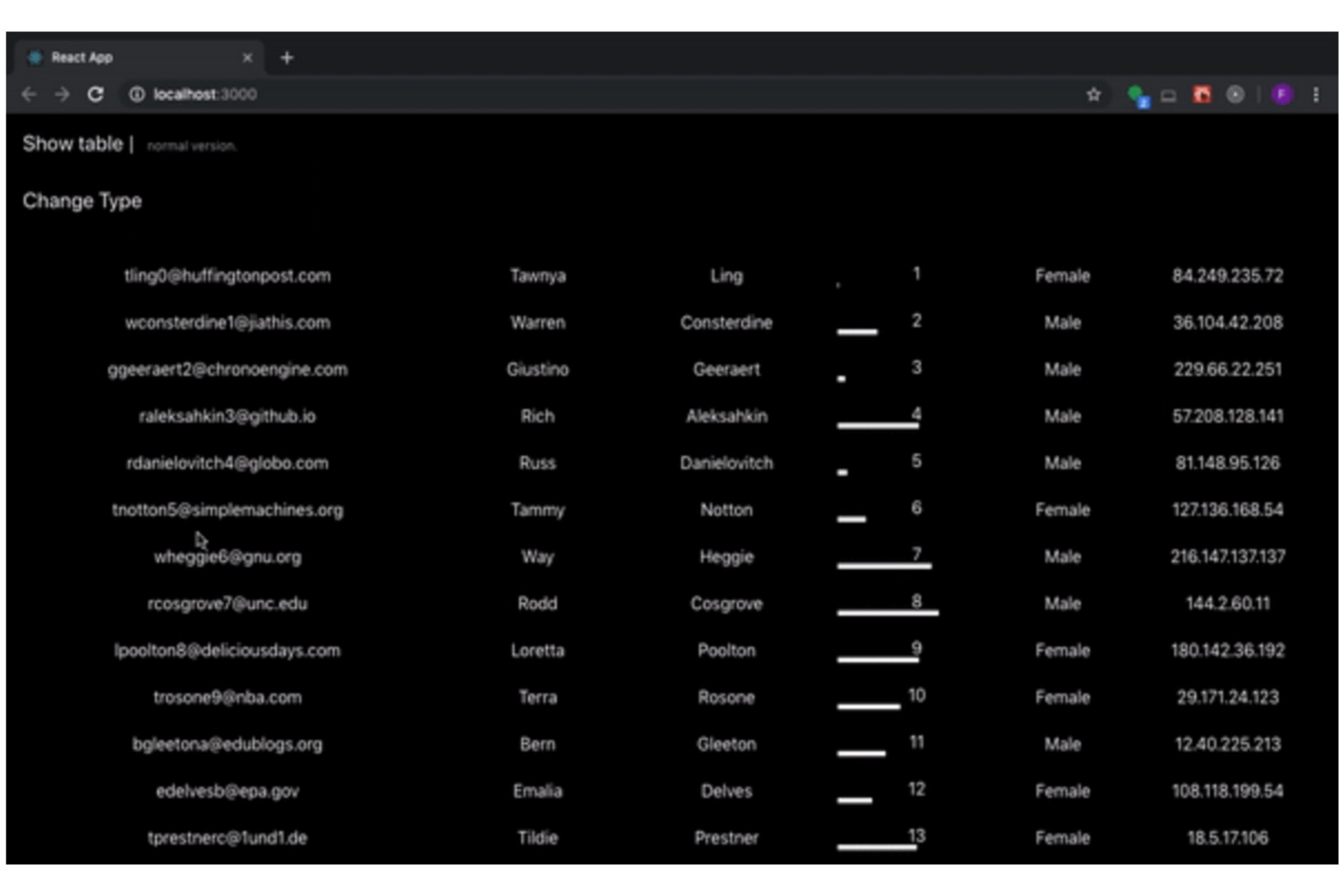
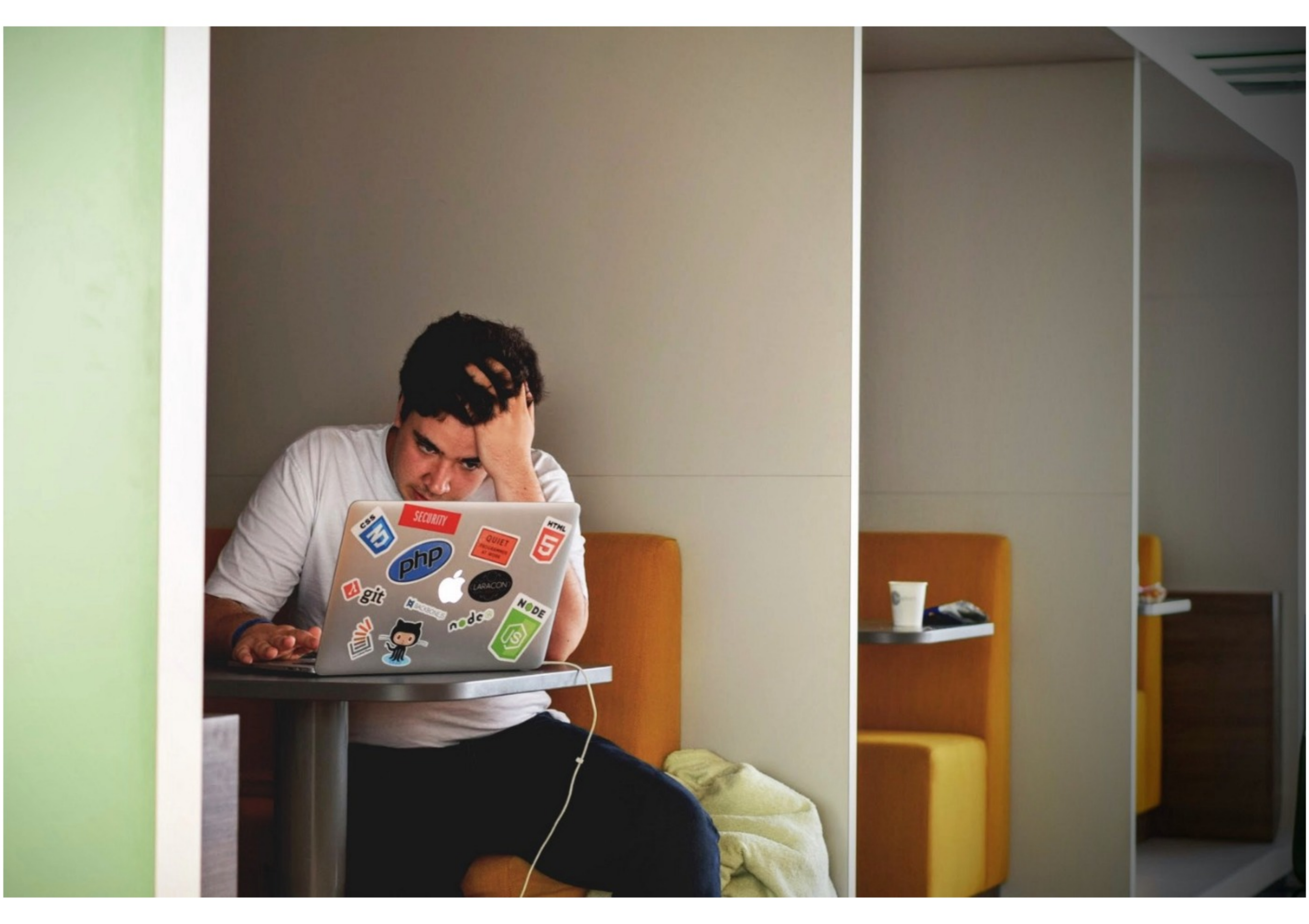
Build an infinite scroll table without scroll event listener.

Frank Tsai Follow
Jun 13 · 3 min read

Why… it's… so… laggy…



When render a huge table with some transition animation, if we don't take care the rendering effort, we are not able to deliver a UX friendly application.



In most of the case, we use event listener to build the infinite scroll table, we not only need to do the calculation between the viewport top and the row height, but also have to write lots of logic in the scroll handler, to prevent frequently re-render, due to each scroll callback will trigger setState function.

The code will be like:

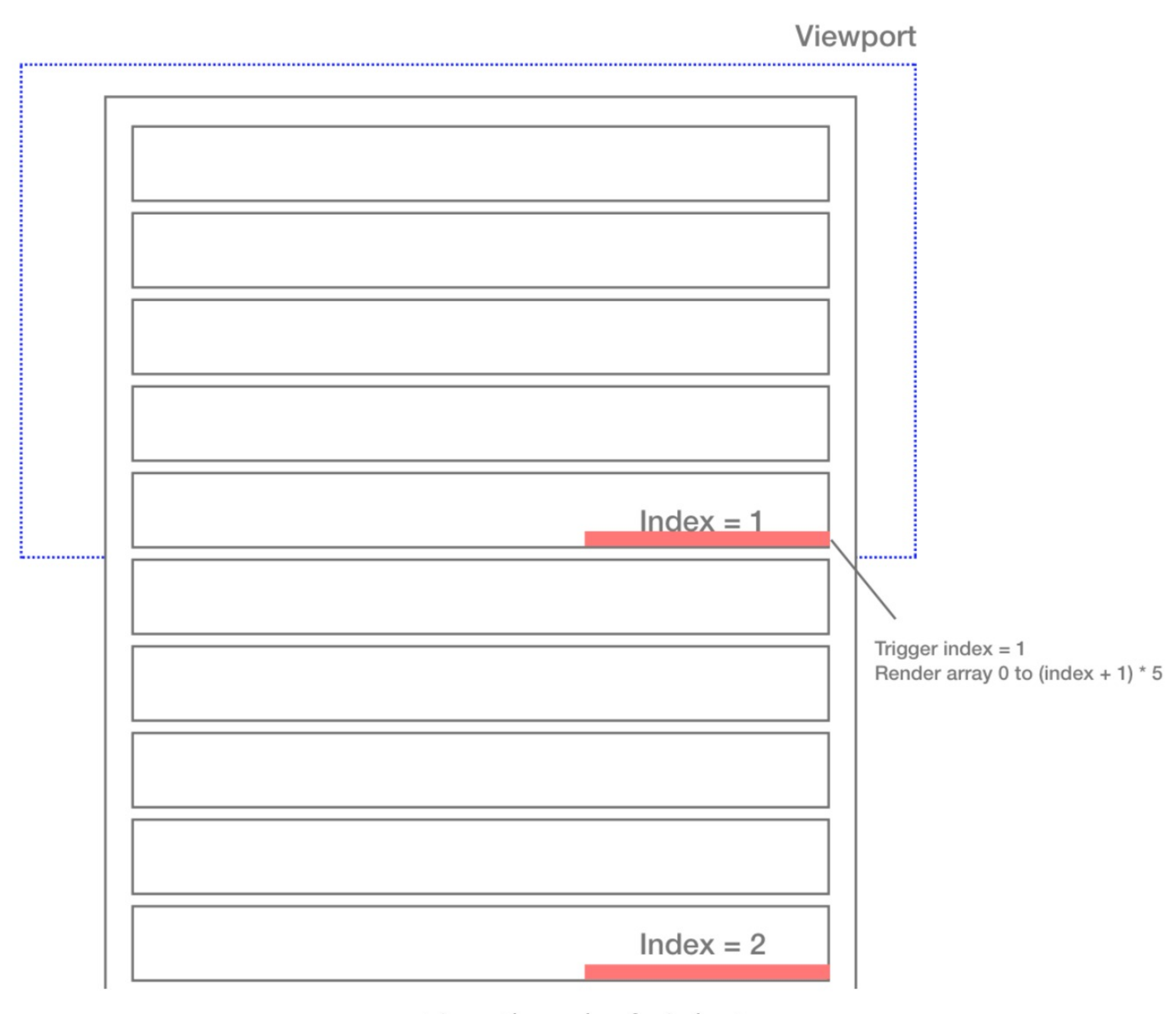
```
componentDidMount() {
  window.addEventListener('scroll', this.handleScroll)
}

handleScroll(e) {
  // use window offset and boundingRect
  const { ...someAttributes } = window;
  const { ...someBoundingRect } = this.component
  // some logic prevent re-render
  if ( ... ) return;
  // do some math
  const newIndex = ...
  // and how many rows should be rendered
  this.setState({index: newIndex })
}
```

There is another way to implement infinite scroll table, without knowing any value of window and component boundingRect.
It is **IntersectionObserver**. The definition from w3c:

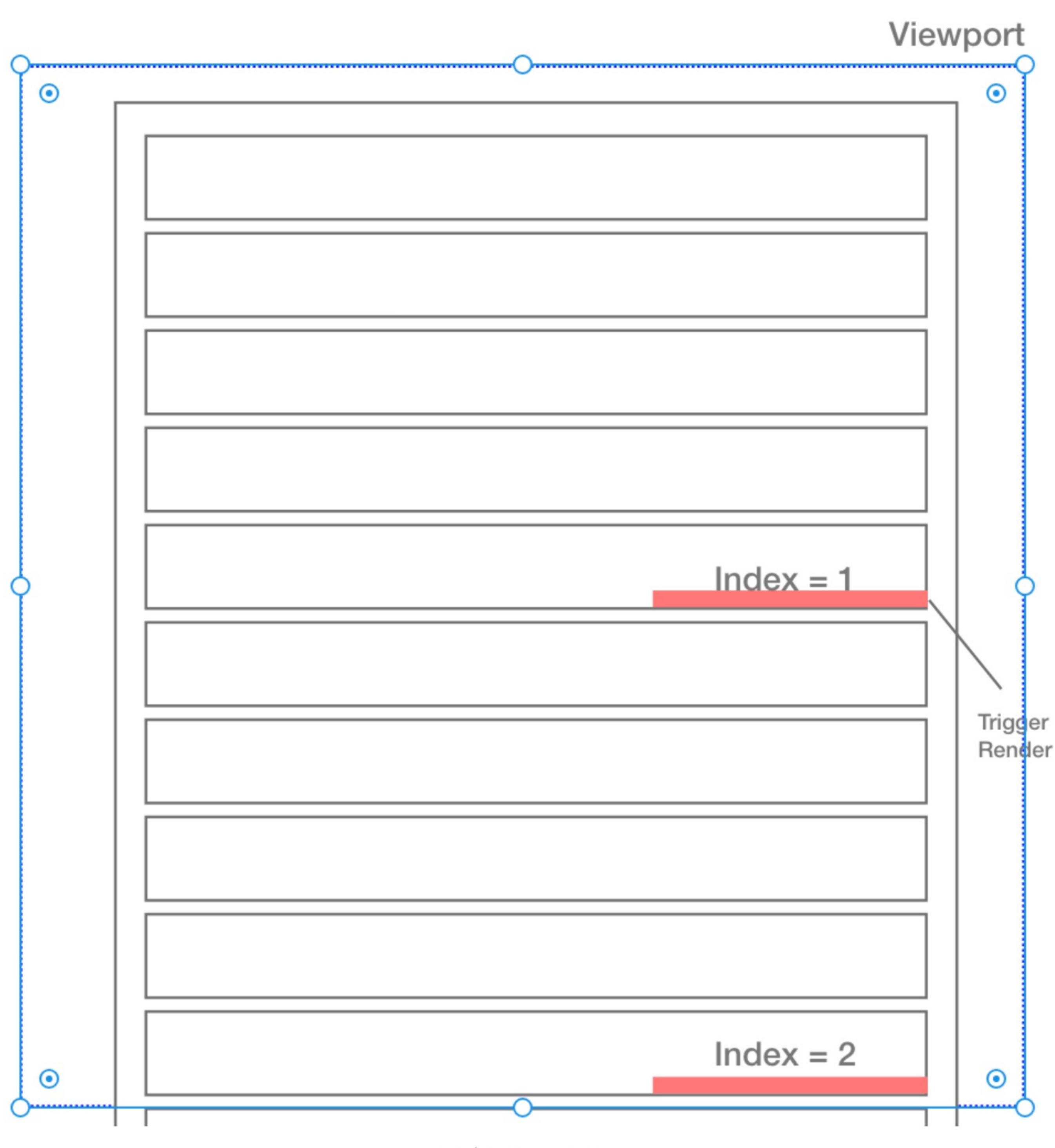
> *This specification describes an API that can be used to understand the visibility and position of DOM elements ("targets") relative to a containing element*

With this implementation, you don't even need to know row height, current viewport top, or any other value to do the math.

The concept is to insert anchors with index attribute in each checkpoint, every time there is an anchor triggered, get the index value and re-render the table. So we don't need to do some magic math with the DOM height and viewport.



trigger the anchor for index 1



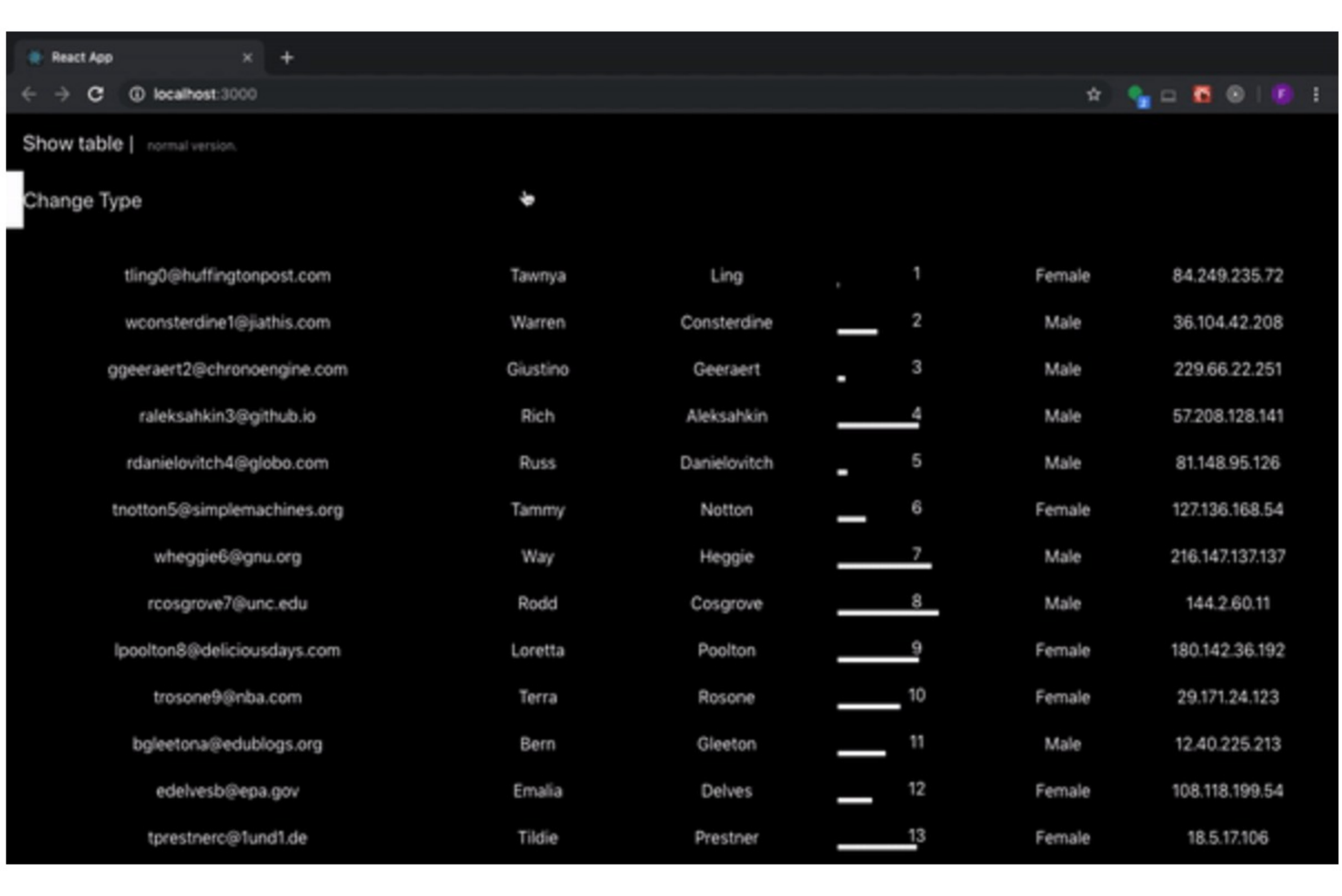
render more rows

Using IntersectionObserver would be like.

```
handleSentinel = (c) => {

  if(!this.observer) {
    // create observer
    this.observer = new IntersectionObserver(
      entries => {
        entries.forEach(e => {
          // if the anchor is triggered, render next section
          if (e.isIntersecting) {
            this.setState(
              { cursor: +e.target.getAttribute('index') }
            );
          }
        });
      },
      {
        root: document.querySelector('App'),
        rootMargin: '-30px',
      }
  }
  if (!c) return;
  // observe new anchor
  this.observer.observe(c)
}

render() {
  const blockNum = 5;
  return(
  ...
  <tbody>
    {MOCK_DATA.slice(0, (cursor+1) * blockNum).map(d =>
      <Block>
        {
          // insert anchor in each checkpoint
          // put an anchor every 5 rows
          d.id % blockNum === 0 ?
          <span ref={this.handleSentinel} index={d.id / blockNum} />
          : null
        }
    </Block>)}
  </tbody>
  ...
  )
}
```

Full code here

With infinite scroll



· · ·

Clap if you like it and follow me for more interested article!

Thanks for reading! :)

You can also find me on LinkedIn, Instagram, Facebook, Github.

Frank Tsai Follow
Software Engineer@Appier | https://mhtsai.me | Find the thing you love so much that you don't even want to stop | Put down some thoughts.