Measure, improve, measure again, improve, measure once mo

```
{data.map((dataPoint, index) => {
  const className = classNames("HeatMap__DataPoint", {
    "HeatMap__DataPoint-isActive": activeDataPoint === index,
  });
  return (
    <div
      tabIndex="0"
      key={index}
      onMouseEnter={(evt) => handleInteraction(index, evt)}
      onFocus={() => handleInteraction(index)}
      onMouseLeave={() => handleInteraction(null)}
      onBlur={() => handleInteraction(null)}
      style={{
        width: blockWidth,
        backgroundColor: colorScale(dataPoint.number),
      }}
      className={className}
      aria-labelledby={tooltipId}
    ></div>
  );
})}
```

Measure, improve, measure again, improve, measure once mo

```
{data.map((dataPoint, index) => {
  const isActive = state.activeDataPoint === index;
  return (
    <Cell
      key={`${dataPoint.time}-${dataPoint.rawValue}`}
      backgroundColor={colorScale(dataPoint.rawValue)}
      onMouseEnter={handleMouseEnter}
      index={index}
      onFocus={handleFocus}
      onBlur={resetTooltip}
      isActive={isActive}
      height={height}
      width={blockWidth}
      tooltipId={tooltipId}
    />
  );
})}
```

```
{data.map((dataPoint, index) => {
  const className = classNames("HeatMap__DataPoint", {
    "HeatMap__DataPoint-isActive": activeDataPoint === index,
  });
  return (
    <div
      tabIndex="0"
      key={index}
      onMouseEnter={(evt) => handleInteraction(index, evt)}
      onFocus={() => handleInteraction(index)}
      onMouseLeave={() => handleInteraction(null)}
      onBlur={() => handleInteraction(null)}
      style={{
        width: blockWidth,
        backgroundColor: colorScale(dataPoint.number),
      }}
      className={className}
      aria-labelledby={tooltipId}
    ></div>
  );
})}
```