Graphic = fn(state)

Romello Goodman

romellogoodman.com/graphic-function-state

My names Romello 👋🏿

- By Day, I'm a Sr Software Engineer at The New York Times 📰
- My team helps people understand the news with new formats (Live Blog) and pages (Covid Hub)
- We're Hiring: ✨ nytco.com/careers ✨

My names Romello 👋🏿

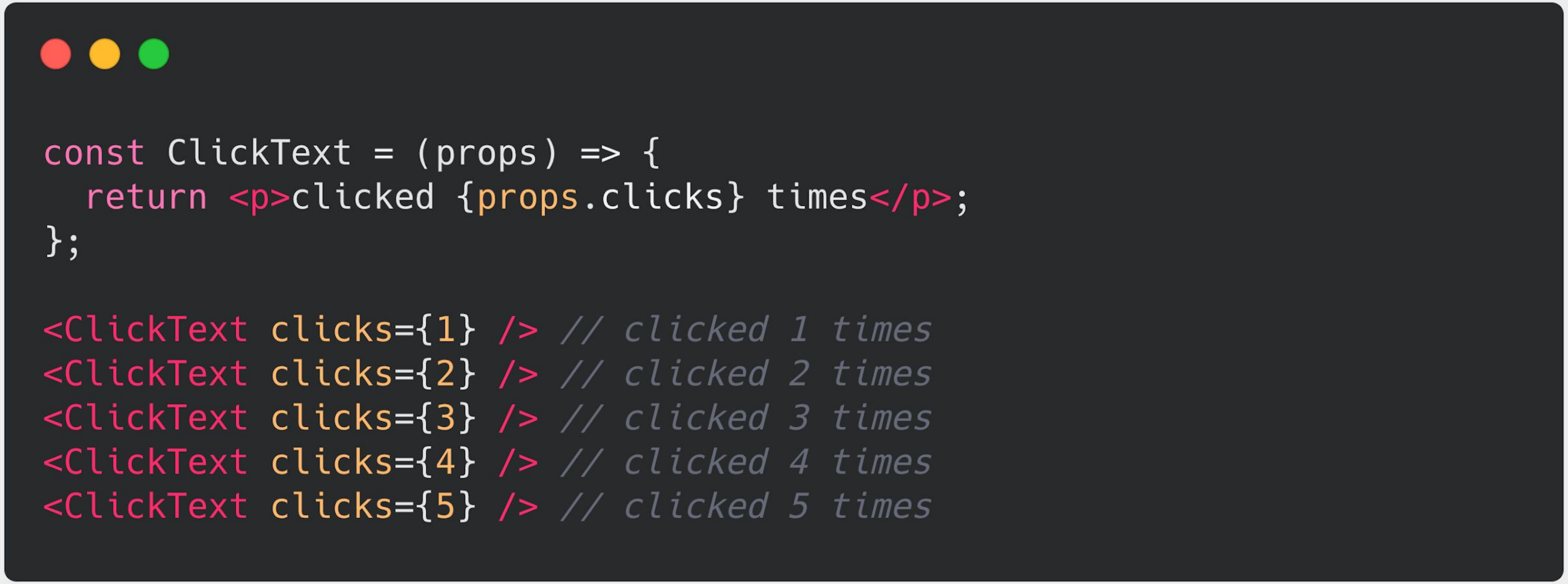
- By Night, I’m a creative coder
- I ❤️ the web
- I am exploring how creativity and code intersect
- Twitter @mellogood
- My latest project is goodgraphics.xyz

Overview

- UI = fn(state)
- SVGs
- SVGs + JSX = <SVGs />
- Graphic Designs can be dynamic and reactive

UI = fn(state)

- In React, your UI is a function of the your data (state)
- Your UI is the rendering of the current state of your data
- As your state updates and changes, your UI updates
- With JSX we can declaratively write our markup

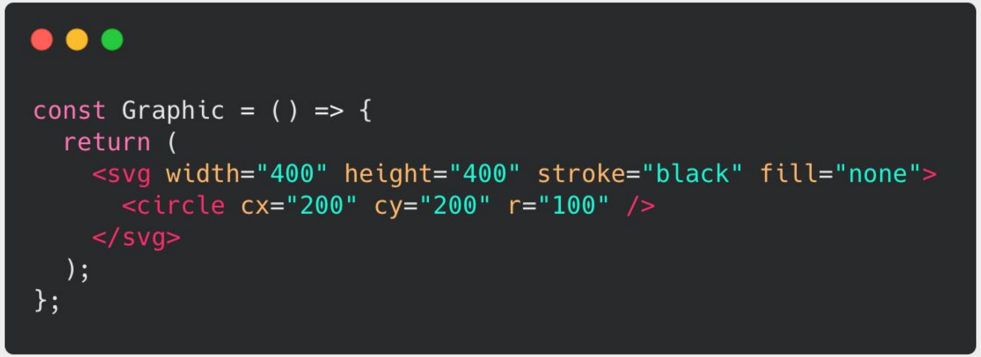


```
const ClickText = (props) => {
  return <p>clicked {props.clicks} times</p>;
};

<ClickText clicks={1} /> // clicked 1 times
<ClickText clicks={2} /> // clicked 2 times
<ClickText clicks={3} /> // clicked 3 times
<ClickText clicks={4} /> // clicked 4 times
<ClickText clicks={5} /> // clicked 5 times
```

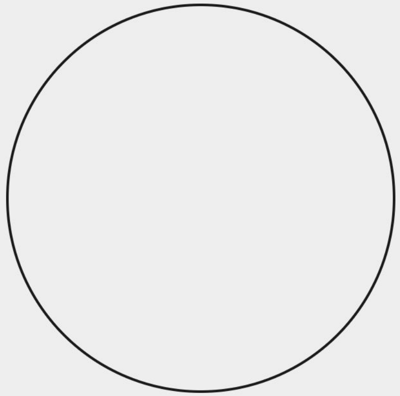
romellogoodman.com/graphic-function-state

SVGs, or Scalable Vector Graphics

- A XML-based markup language for describing 2D based vector graphics [1]
- It’s natively supported by JSX
- The markup looks like React components with special props



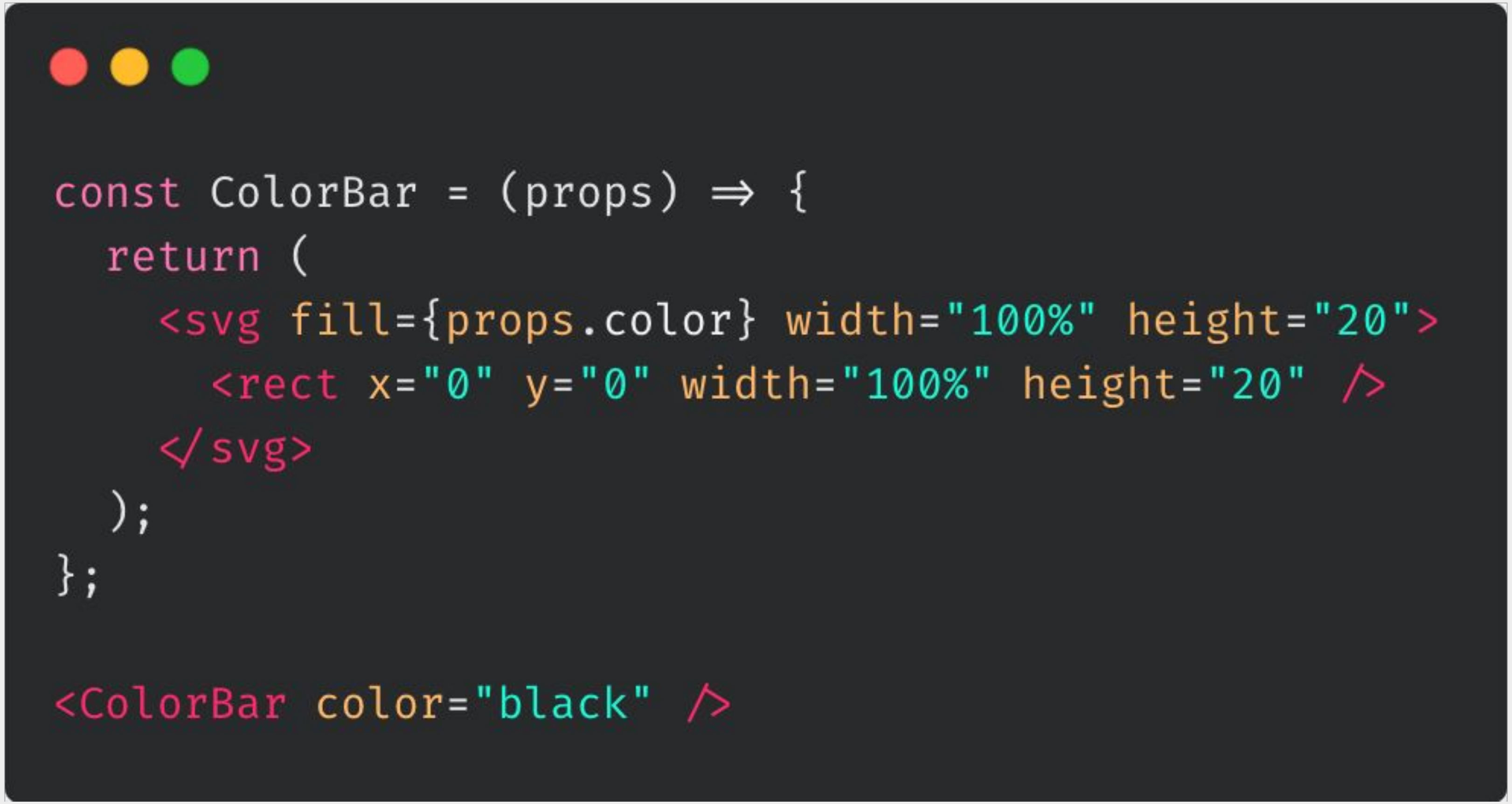
```
const Graphic = () => {
  return (
    <svg width="400" height="400" stroke="black" fill="none">
      <circle cx="200" cy="200" r="100" />
    </svg>
  );
};
```



romellogoodman.com/graphic-function-state

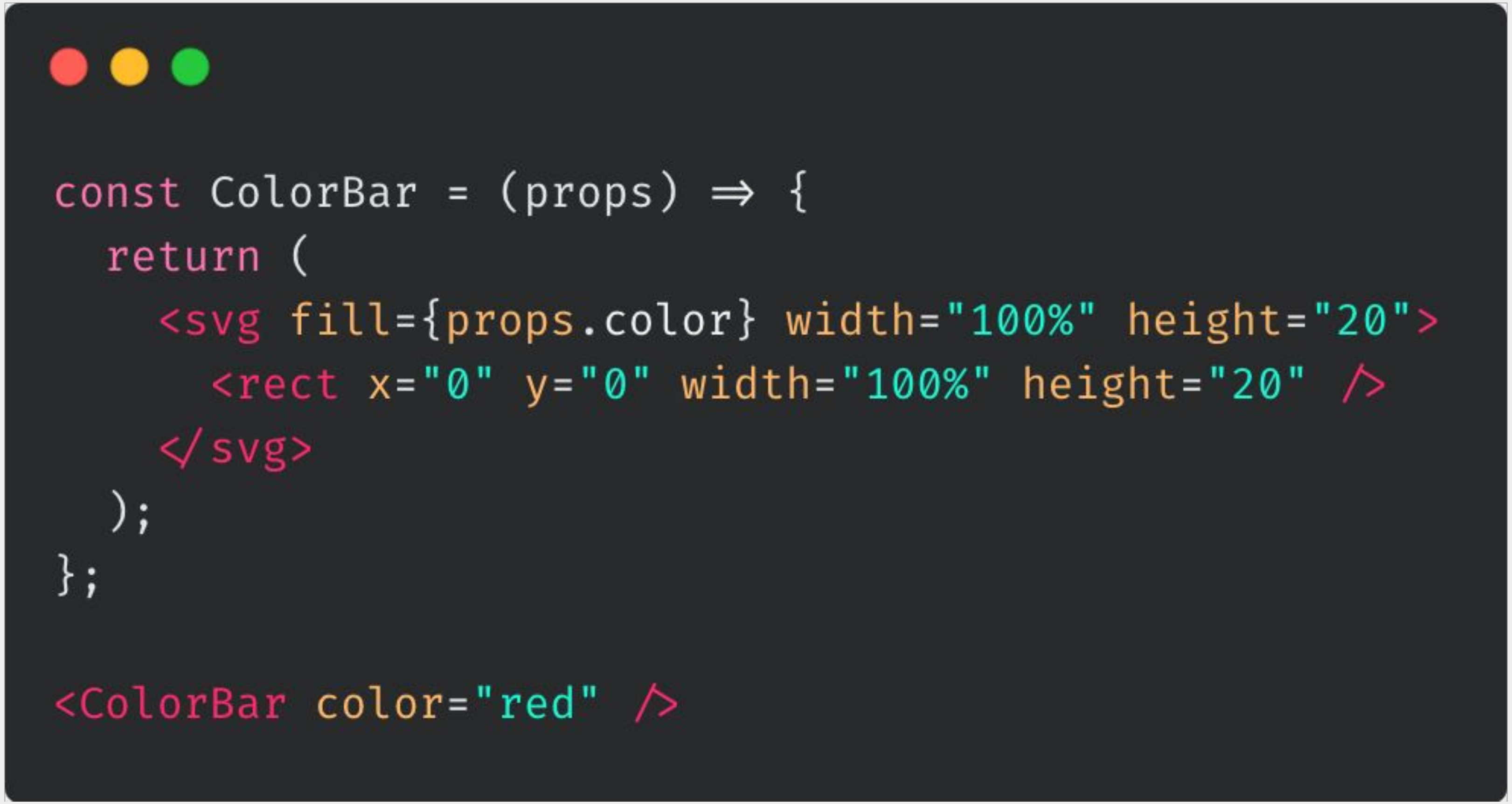
<SVGs />

- <SVGs /> are components
- Components can contain logic and state
- They can be updated based on that state
- JSX + SVGs allow us to declaratively write Graphics



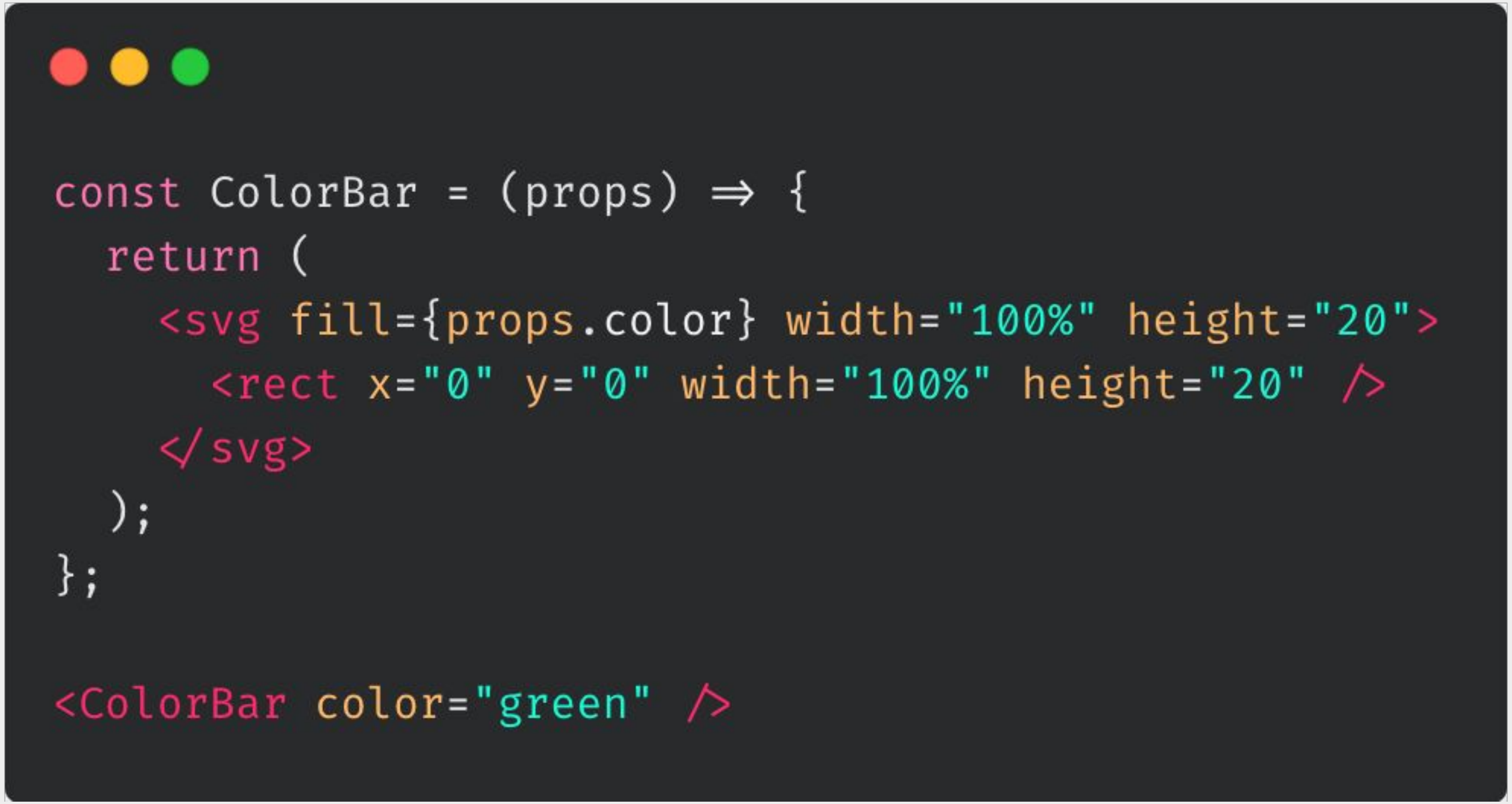
```jsx
const ColorBar = (props) => {
  return (
    <svg fill={props.color} width="100%" height="20">
      <rect x="0" y="0" width="100%" height="20" />
    </svg>
  );
};

<ColorBar color="black" />
```

romellogoodman.com/graphic-function-state



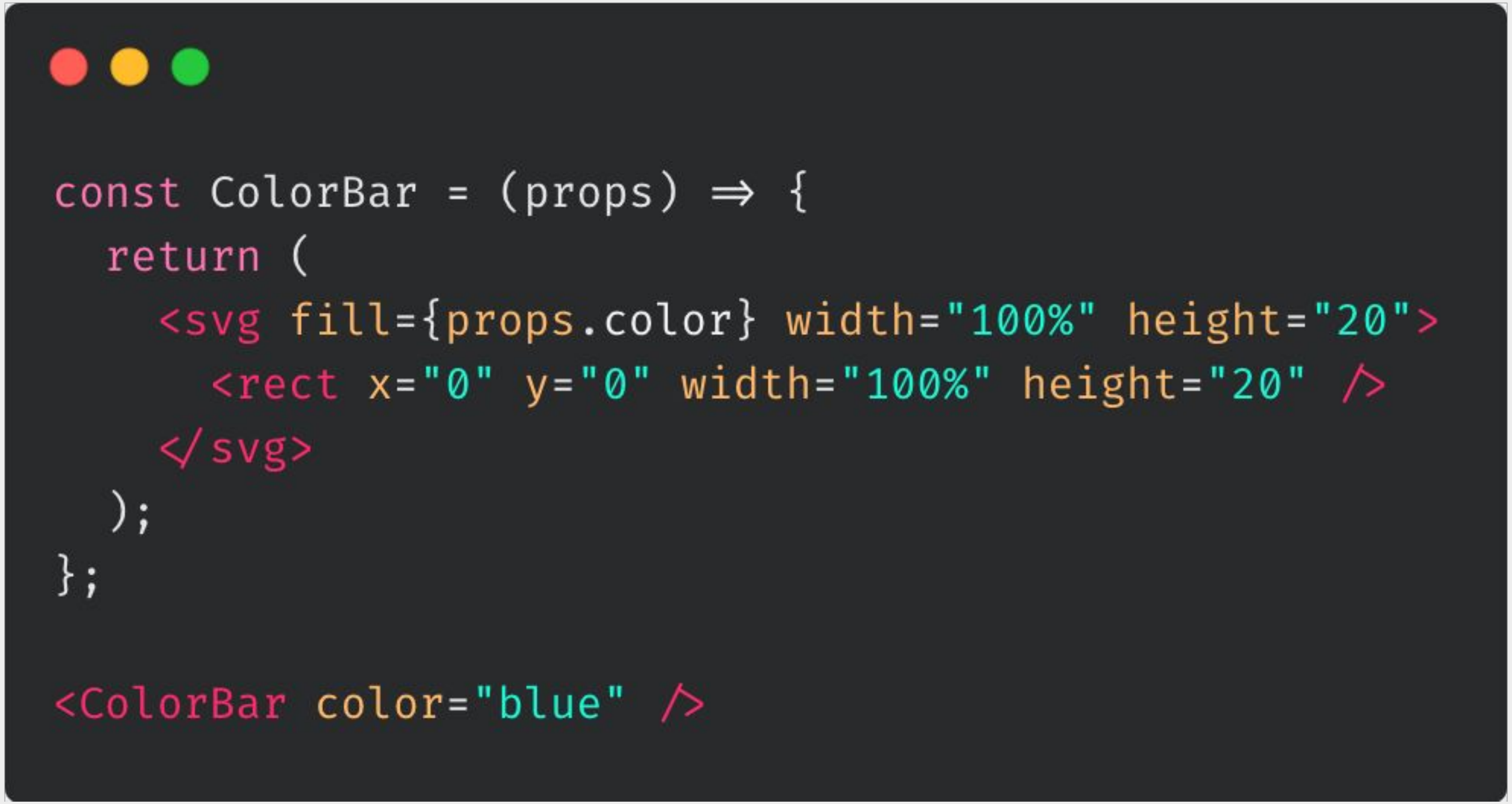
```
const ColorBar = (props) => {
  return (
    <svg fill={props.color} width="100%" height="20">
      <rect x="0" y="0" width="100%" height="20" />
    </svg>
  );
};

<ColorBar color="red" />
```

romellogoodman.com/graphic-function-state



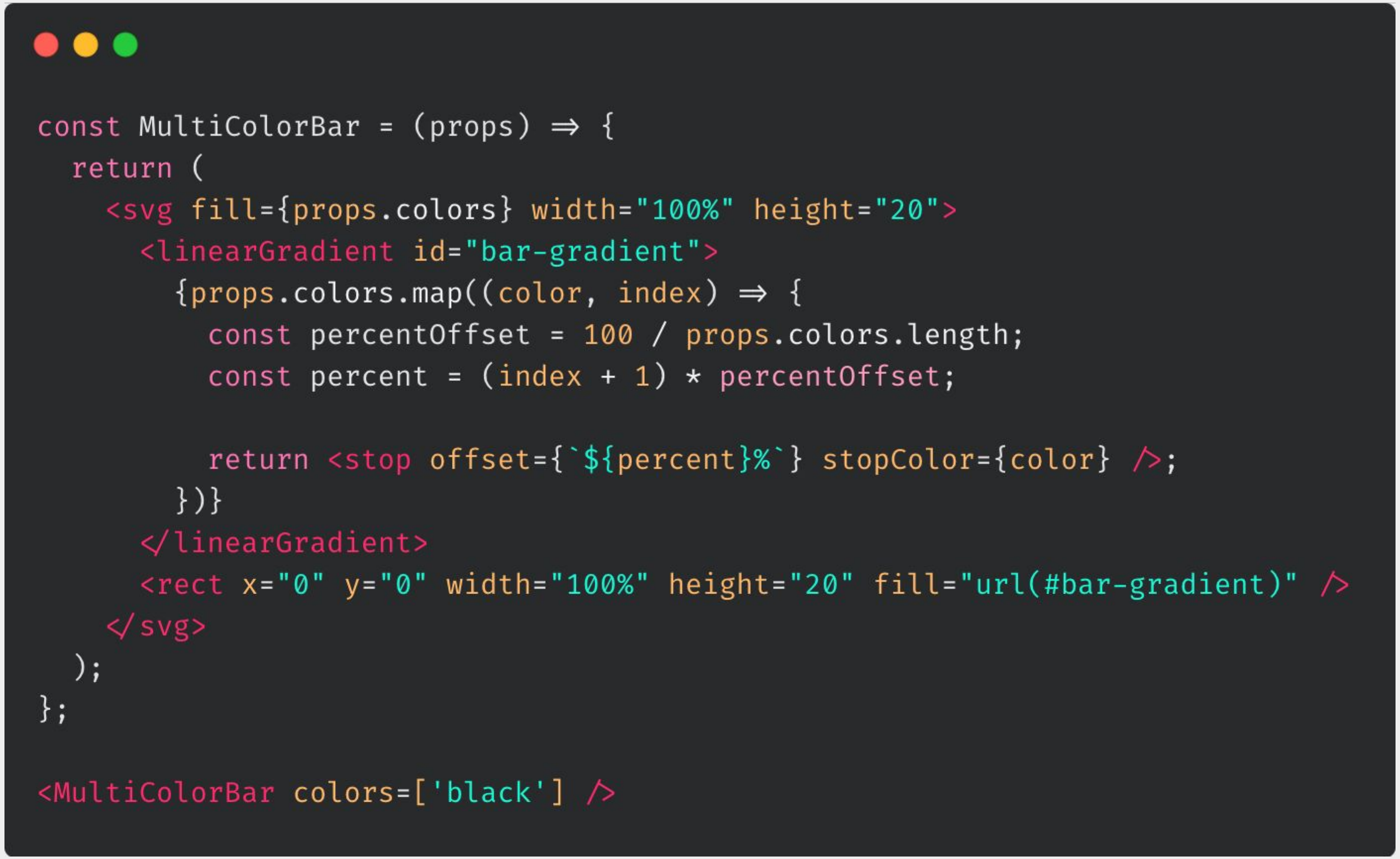
```jsx
const ColorBar = (props) => {
  return (
    <svg fill={props.color} width="100%" height="20">
      <rect x="0" y="0" width="100%" height="20" />
    </svg>
  );
};

<ColorBar color="green" />
```

romellogoodman.com/graphic-function-state



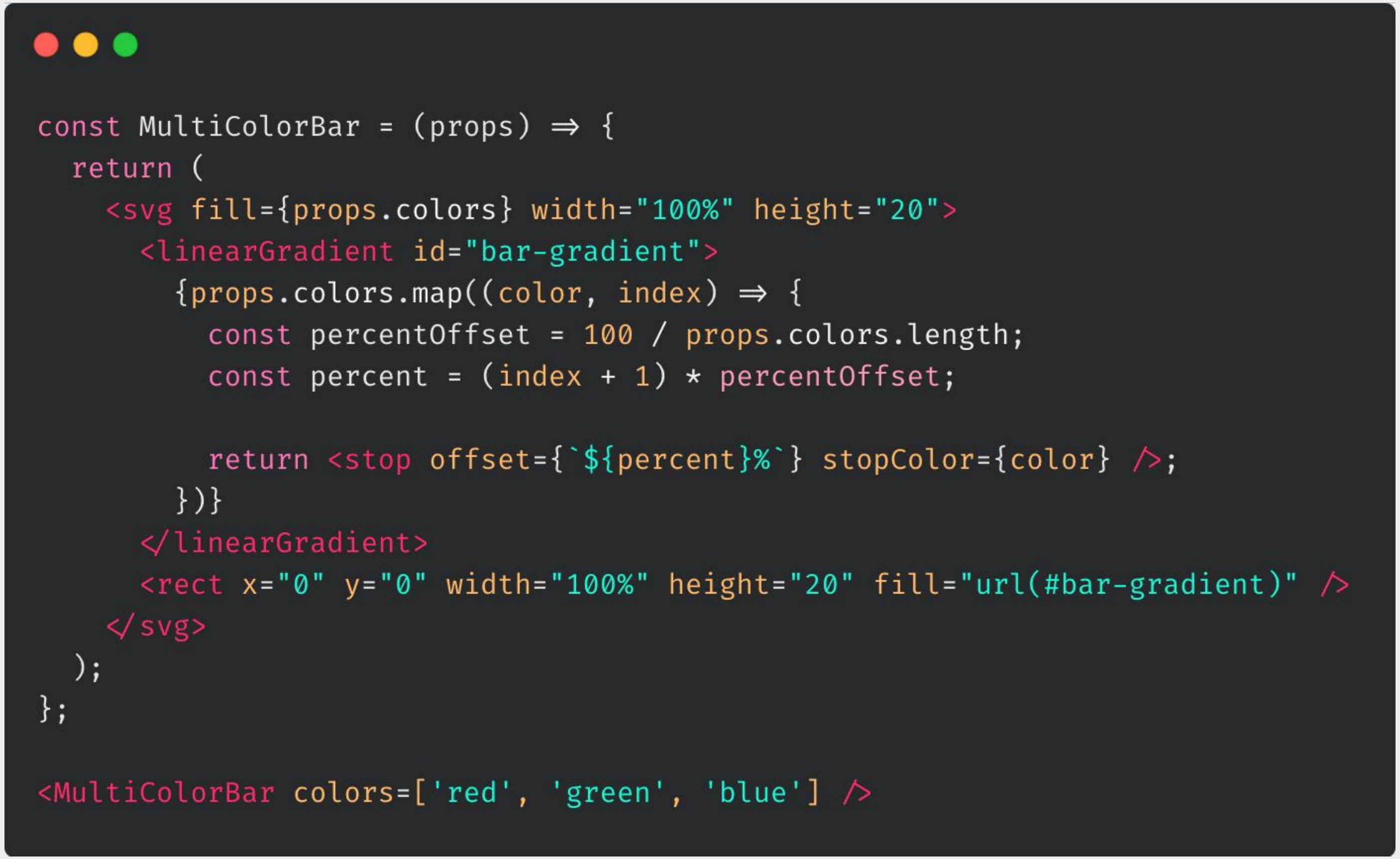
```
const ColorBar = (props) => {
  return (
    <svg fill={props.color} width="100%" height="20">
      <rect x="0" y="0" width="100%" height="20" />
    </svg>
  );
};

<ColorBar color="blue" />
```

romellogoodman.com/graphic-function-state



```
const MultiColorBar = (props) => {
  return (
    <svg fill={props.colors} width="100%" height="20">
      <linearGradient id="bar-gradient">
        {props.colors.map((color, index) => {
          const percentOffset = 100 / props.colors.length;
          const percent = (index + 1) * percentOffset;

          return <stop offset={`${percent}%`} stopColor={color} />;
        })}
      </linearGradient>
      <rect x="0" y="0" width="100%" height="20" fill="url(#bar-gradient)" />
    </svg>
  );
};

<MultiColorBar colors=['black'] />
```

romellogoodman.com/graphic-function-state

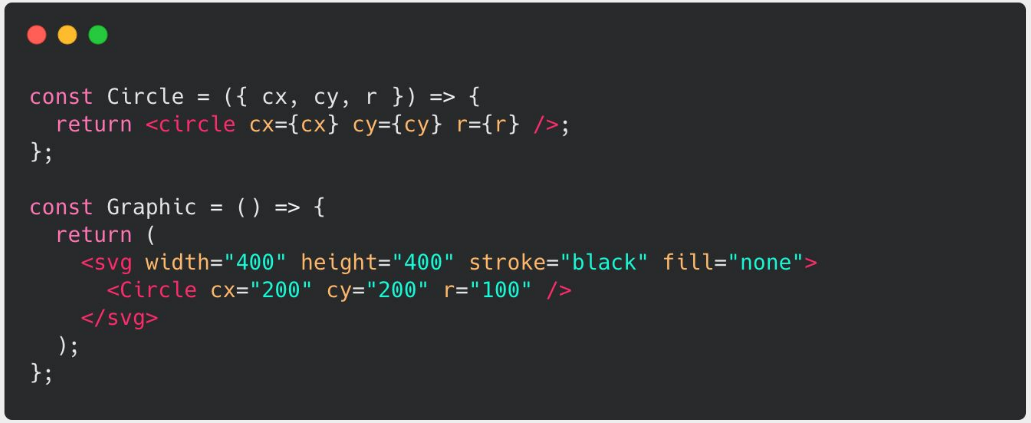


```jsx
const MultiColorBar = (props) => {
  return (
    <svg fill={props.colors} width="100%" height="20">
      <linearGradient id="bar-gradient">
        {props.colors.map((color, index) => {
          const percentOffset = 100 / props.colors.length;
          const percent = (index + 1) * percentOffset;

          return <stop offset={`${percent}%`} stopColor={color} />;
        })}
      </linearGradient>
      <rect x="0" y="0" width="100%" height="20" fill="url(#bar-gradient)" />
    </svg>
  );
};

<MultiColorBar colors=['red', 'green', 'blue'] />
```

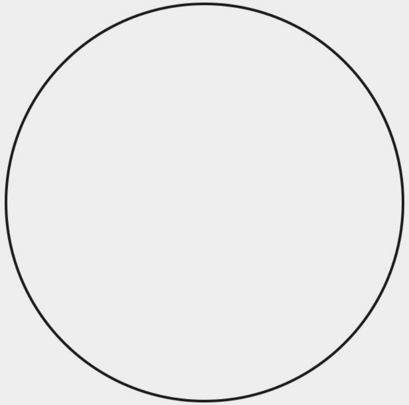
romellogoodman.com/graphic-function-state

Graphic = fn(state)

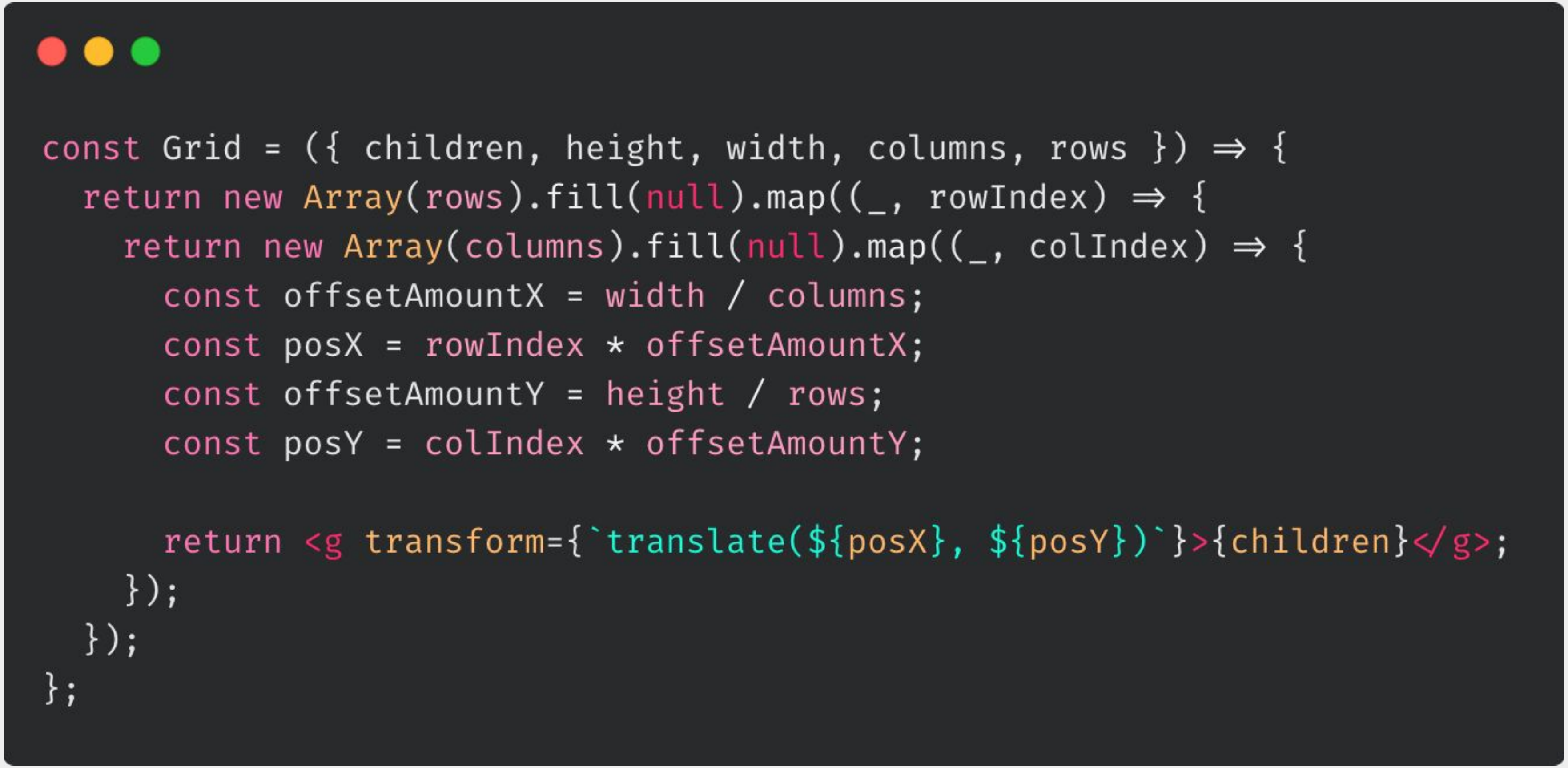
- Using plain HTML we can build Graphic Systems
- Logic can be broken into <Components />
- Graphic Designs can be rich and reactive



```
const Circle = ({ cx, cy, r }) => {
  return <circle cx={cx} cy={cy} r={r} />;
};

const Graphic = () => {
  return (
    <svg width="400" height="400" stroke="black" fill="none">
      <Circle cx="200" cy="200" r="100" />
    </svg>
  );
};
```

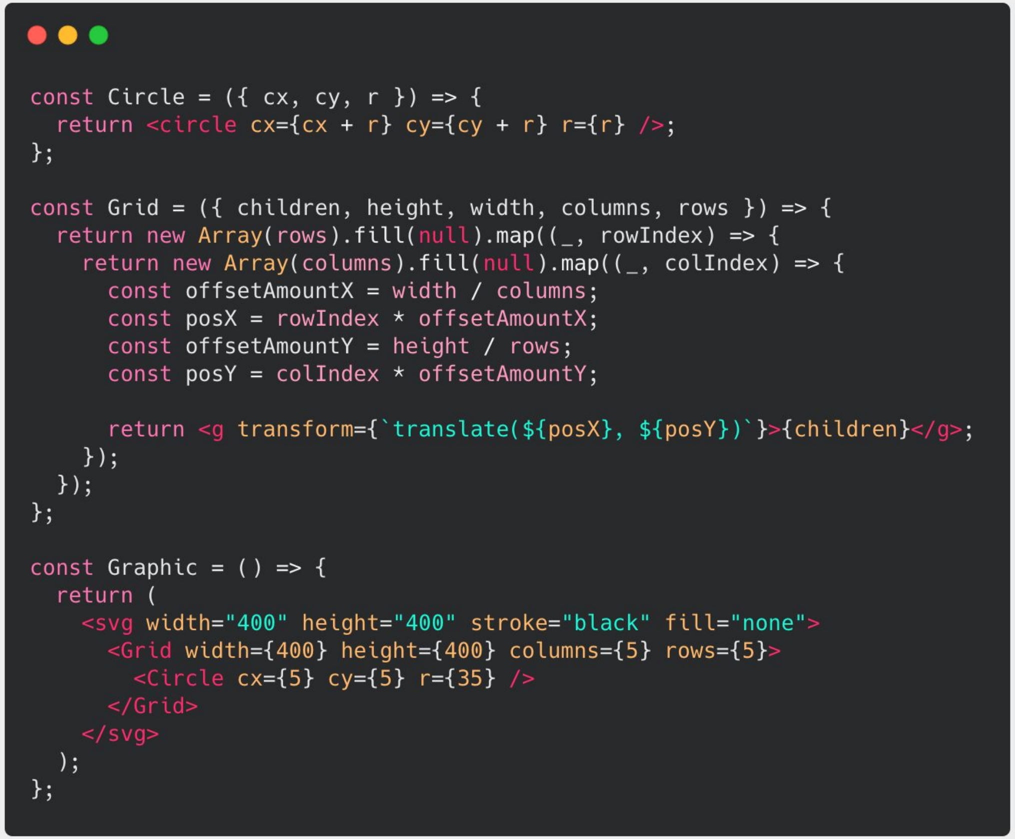


romellogoodman.com/graphic-function-state

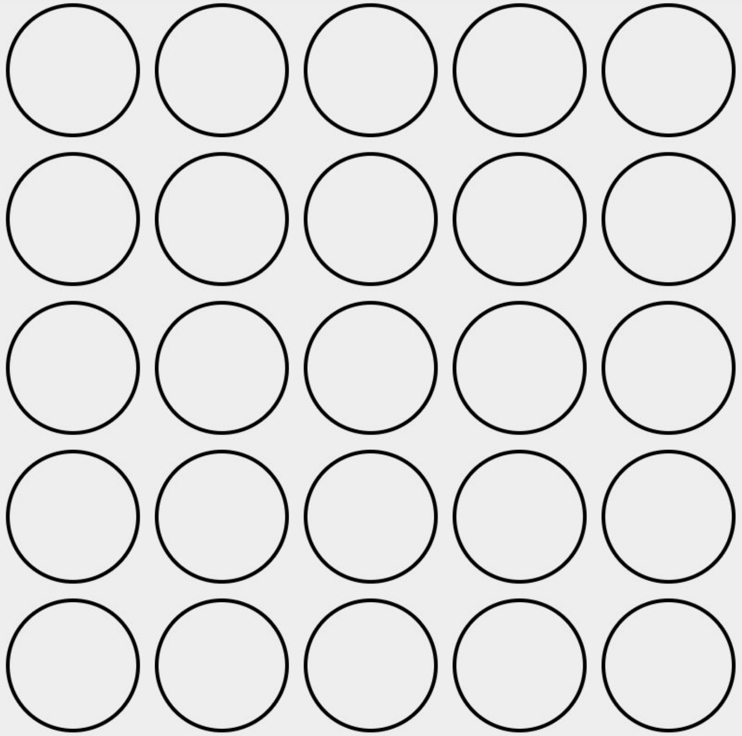


```
const Grid = ({ children, height, width, columns, rows }) => {
  return new Array(rows).fill(null).map((_, rowIndex) => {
    return new Array(columns).fill(null).map((_, colIndex) => {
      const offsetAmountX = width / columns;
      const posX = rowIndex * offsetAmountX;
      const offsetAmountY = height / rows;
      const posY = colIndex * offsetAmountY;

      return <g transform={`translate(${posX}, ${posY})`}>{children}</g>;
    });
  });
};
```

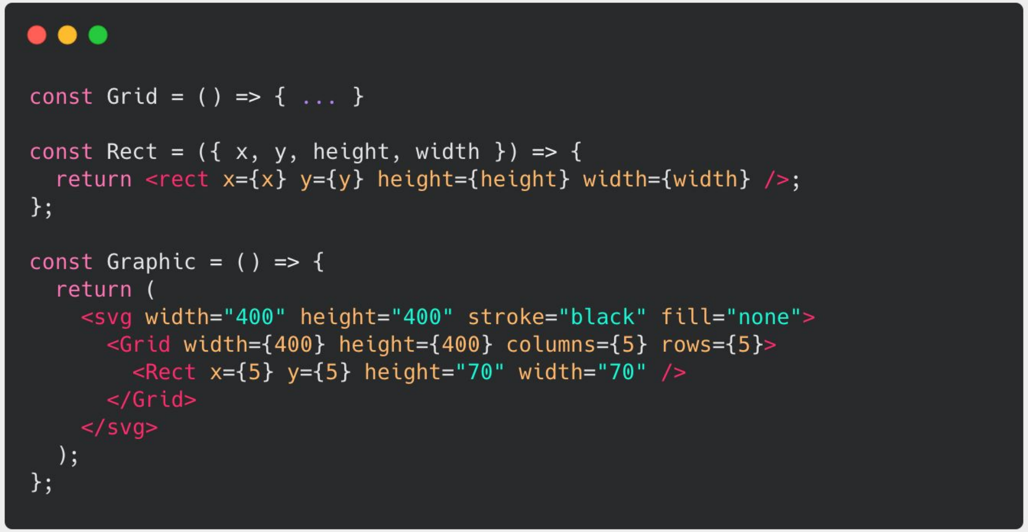
romellogoodman.com/graphic-function-state



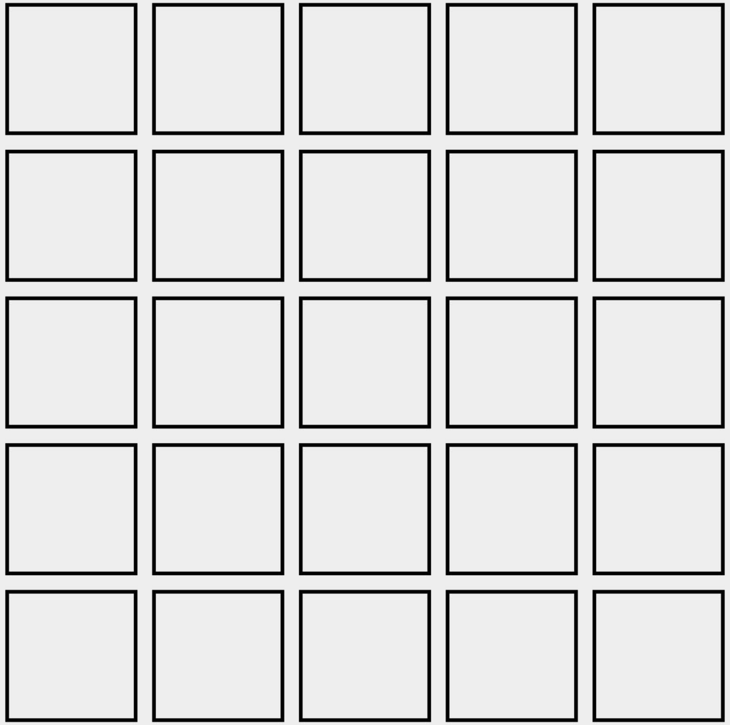
```
const Circle = ({ cx, cy, r }) => {
  return <circle cx={cx + r} cy={cy + r} r={r} />;
};

const Grid = ({ children, height, width, columns, rows }) => {
  return new Array(rows).fill(null).map((_, rowIndex) => {
    return new Array(columns).fill(null).map((_, colIndex) => {
      const offsetAmountX = width / columns;
      const posX = rowIndex * offsetAmountX;
      const offsetAmountY = height / rows;
      const posY = colIndex * offsetAmountY;

      return <g transform={`translate(${posX}, ${posY})`}>{children}</g>;
    });
  });
};

const Graphic = () => {
  return (
    <svg width="400" height="400" stroke="black" fill="none">
      <Grid width={400} height={400} columns={5} rows={5}>
        <Circle cx={5} cy={5} r={35} />
      </Grid>
    </svg>
  );
};
```

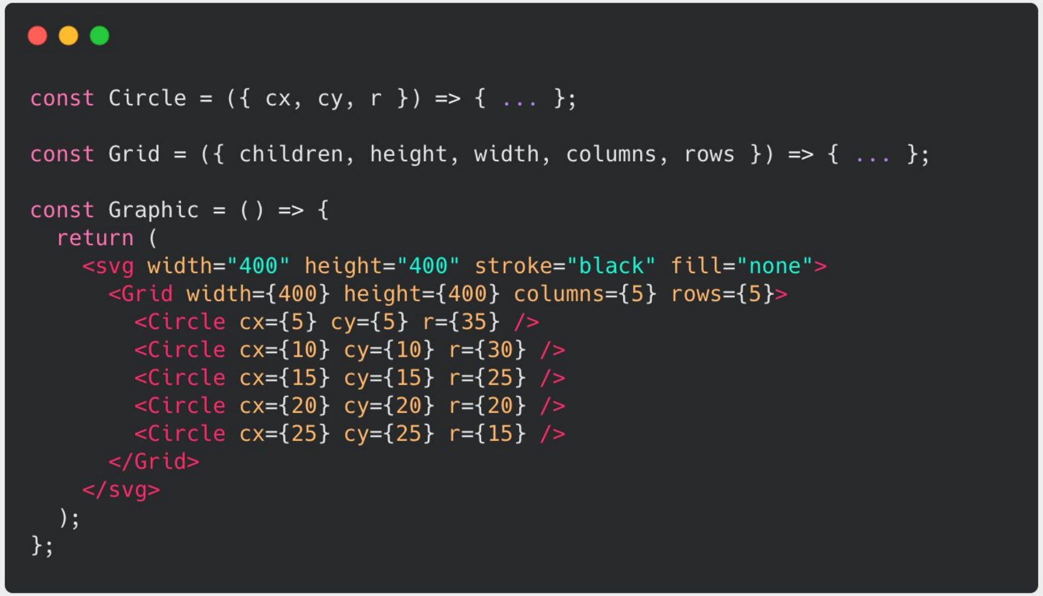


romellogoodman.com/graphic-function-state

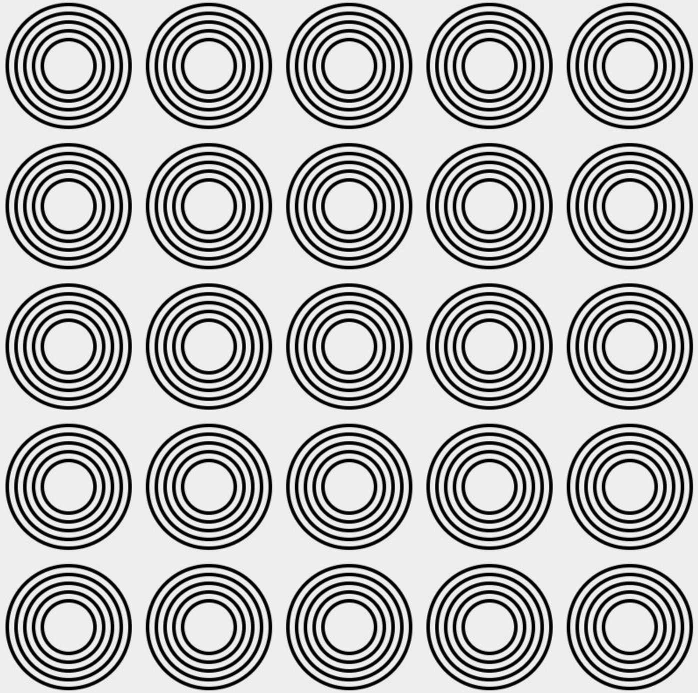


```
const Grid = () => { ... }

const Rect = ({ x, y, height, width }) => {
  return <rect x={x} y={y} height={height} width={width} />;
};

const Graphic = () => {
  return (
    <svg width="400" height="400" stroke="black" fill="none">
      <Grid width={400} height={400} columns={5} rows={5}>
        <Rect x={5} y={5} height="70" width="70" />
      </Grid>
    </svg>
  );
};
```

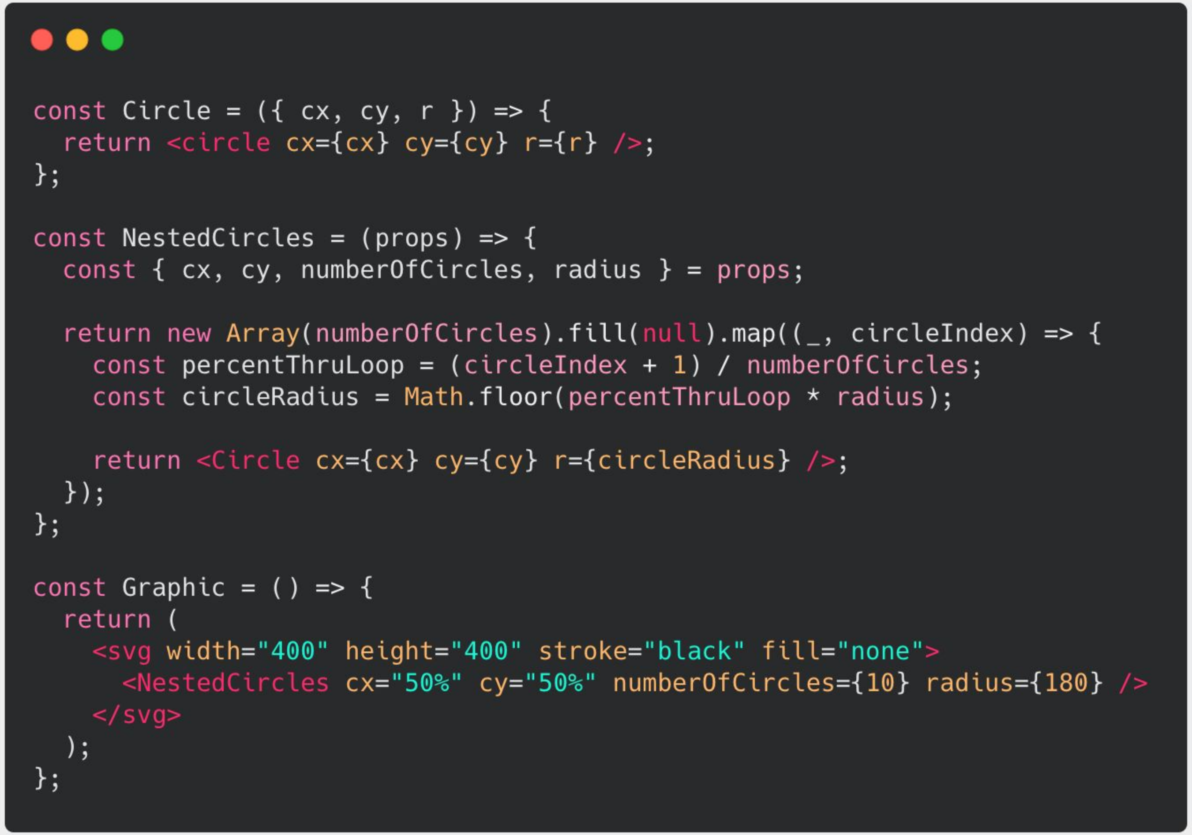




```
const Circle = ({ cx, cy, r }) => { ... };

const Grid = ({ children, height, width, columns, rows }) => { ... };

const Graphic = () => {
  return (
    <svg width="400" height="400" stroke="black" fill="none">
      <Grid width={400} height={400} columns={5} rows={5}>
        <Circle cx={5} cy={5} r={35} />
        <Circle cx={10} cy={10} r={30} />
        <Circle cx={15} cy={15} r={25} />
        <Circle cx={20} cy={20} r={20} />
        <Circle cx={25} cy={25} r={15} />
      </Grid>
    </svg>
  );
};
```

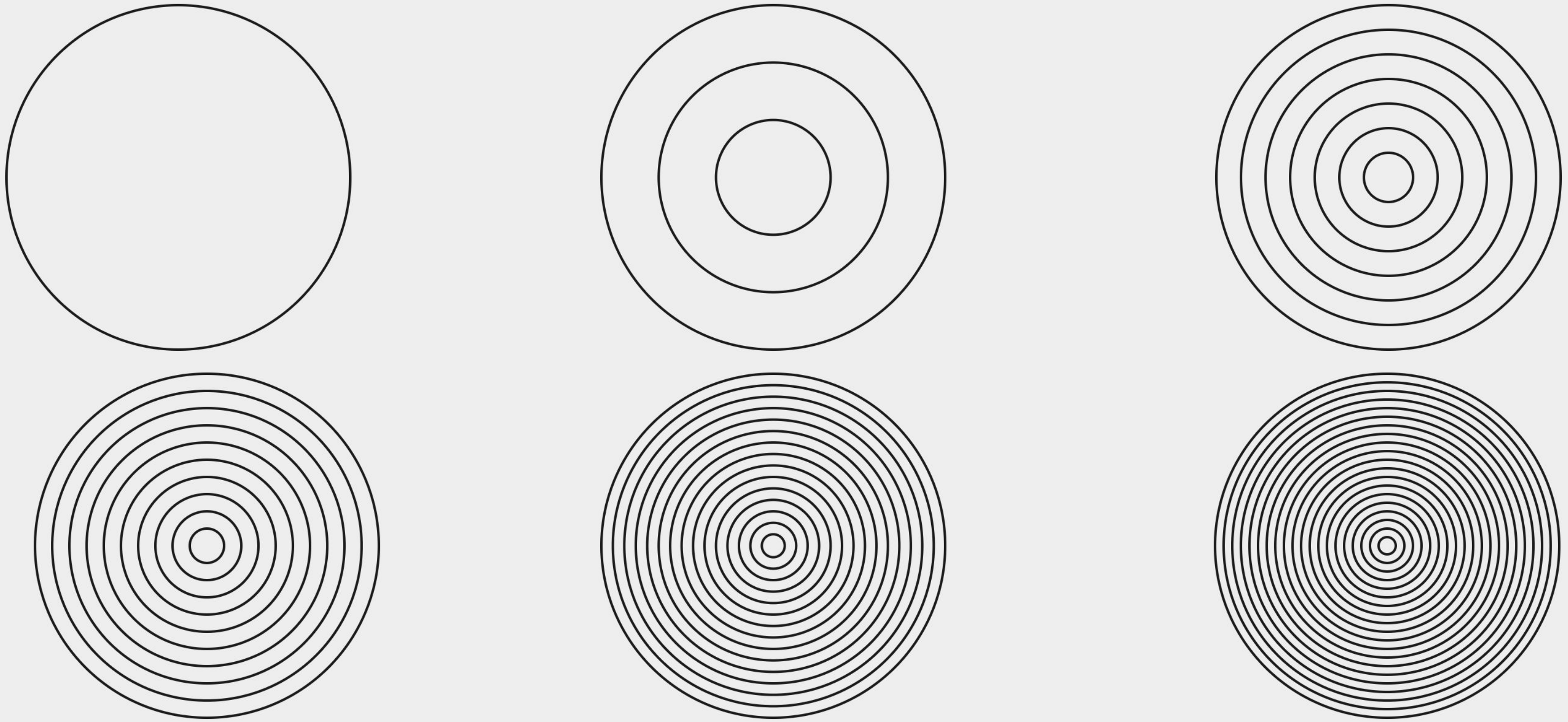


romellogoodman.com/graphic-function-state

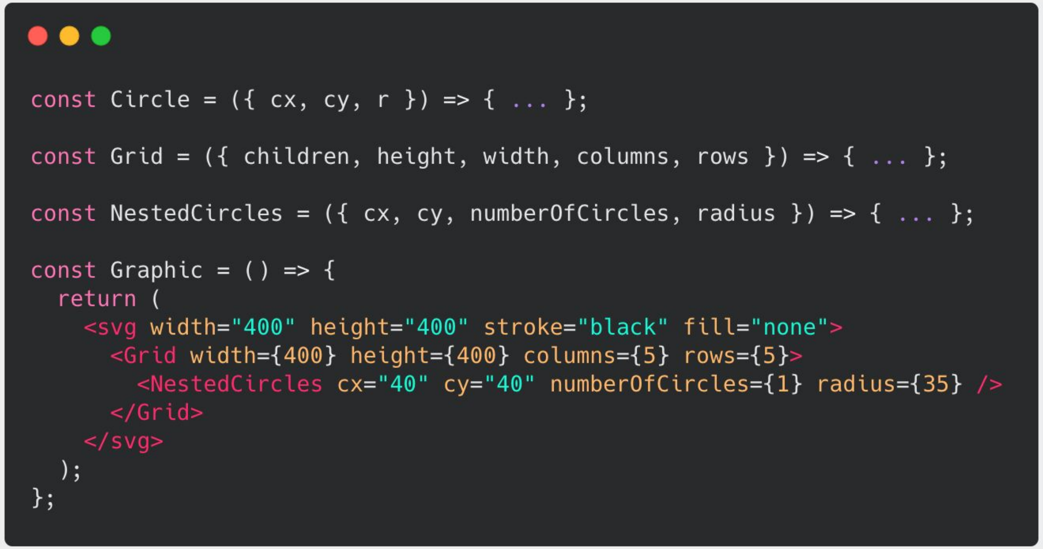


```
const Circle = ({ cx, cy, r }) => {
  return <circle cx={cx} cy={cy} r={r} />;
};

const NestedCircles = (props) => {
  const { cx, cy, numberOfCircles, radius } = props;

  return new Array(numberOfCircles).fill(null).map((_, circleIndex) => {
    const percentThruLoop = (circleIndex + 1) / numberOfCircles;
    const circleRadius = Math.floor(percentThruLoop * radius);

    return <Circle cx={cx} cy={cy} r={circleRadius} />;
  });
};

const Graphic = () => {
  return (
    <svg width="400" height="400" stroke="black" fill="none">
      <NestedCircles cx="50%" cy="50%" numberOfCircles={10} radius={180} />
    </svg>
  );
};
```

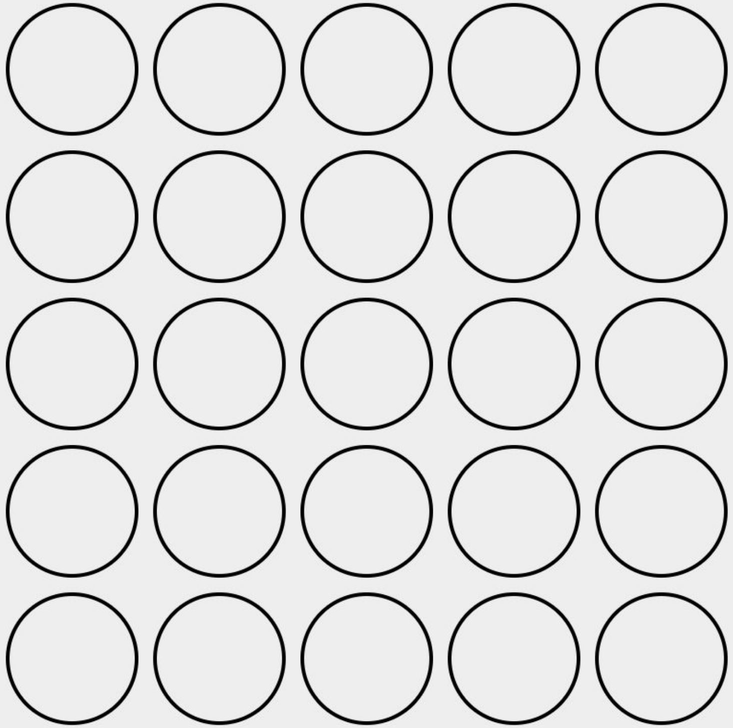
romellogoodman.com/graphic-function-state



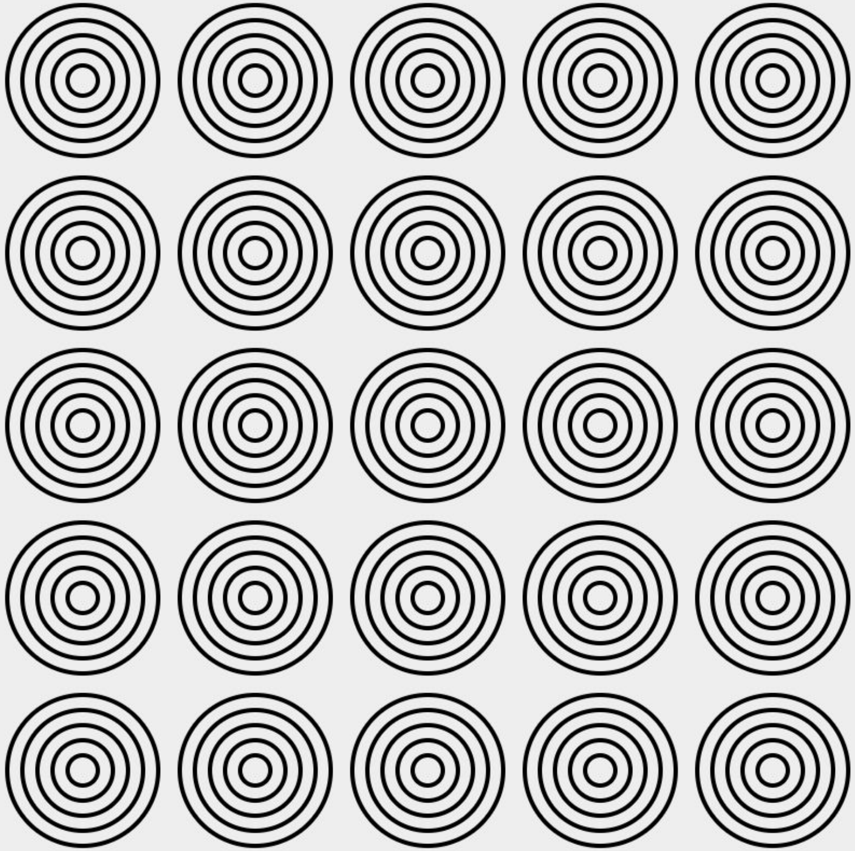
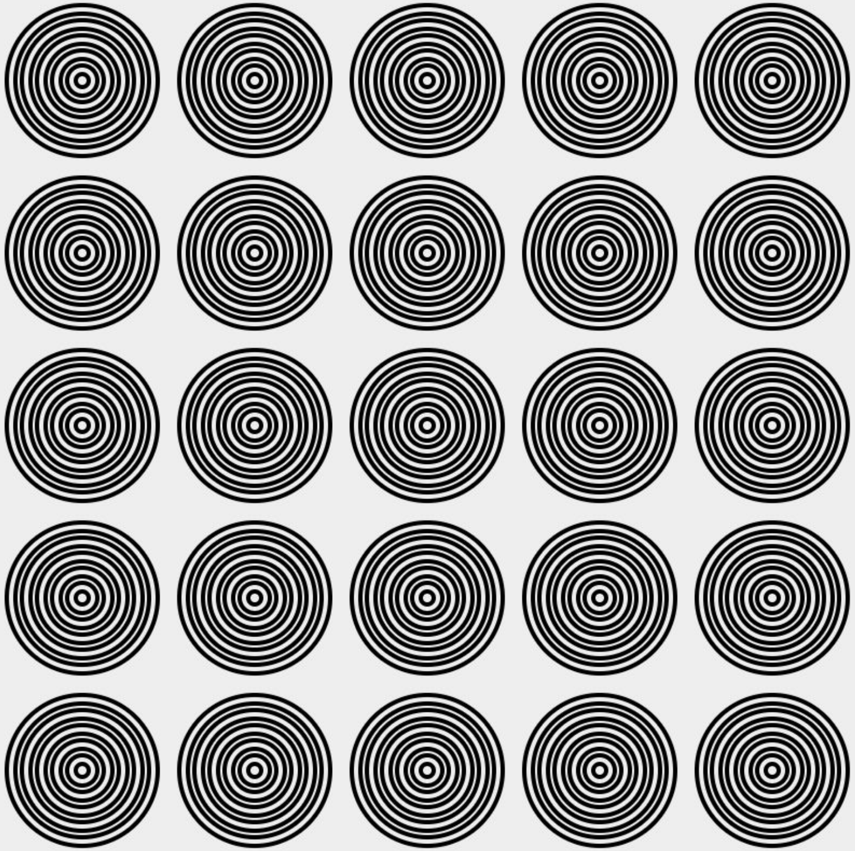
romellogoodman.com/graphic-function-state



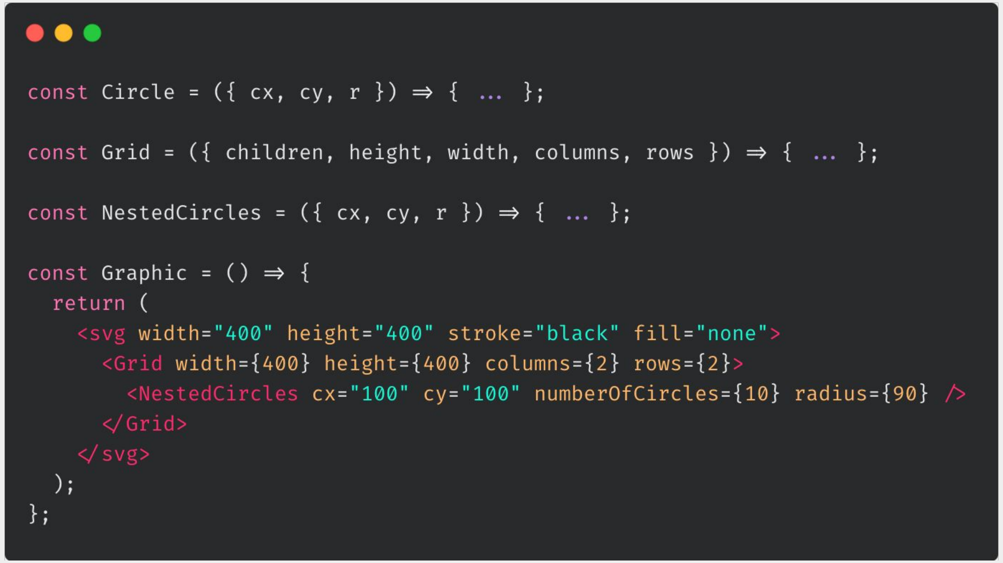
```
const Circle = ({ cx, cy, r }) => { ... };

const Grid = ({ children, height, width, columns, rows }) => { ... };

const NestedCircles = ({ cx, cy, numberOfCircles, radius }) => { ... };

const Graphic = () => {
  return (
    <svg width="400" height="400" stroke="black" fill="none">
      <Grid width={400} height={400} columns={5} rows={5}>
        <NestedCircles cx="40" cy="40" numberOfCircles={1} radius={35} />
      </Grid>
    </svg>
  );
};
```



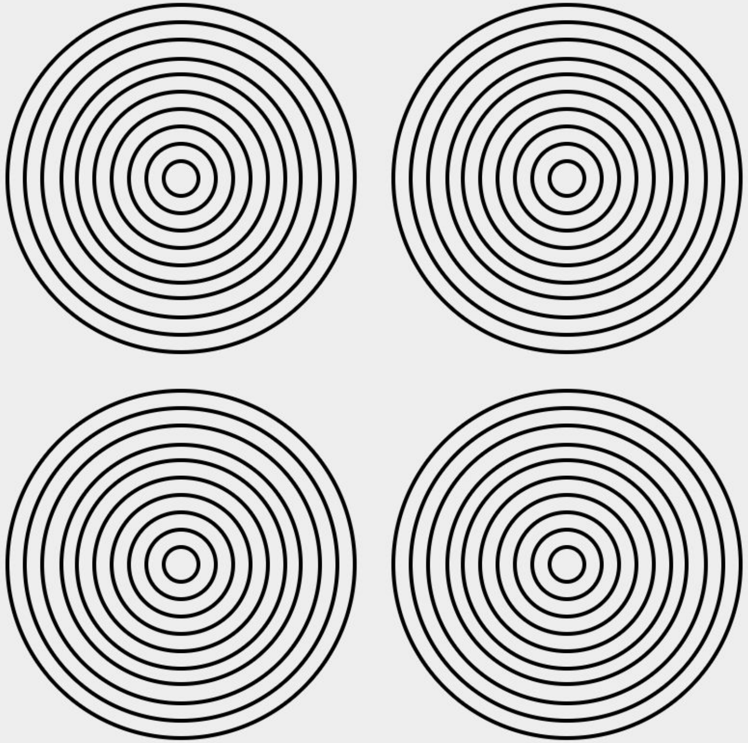
romellogoodman.com/graphic-function-state

romellogoodman.com/graphic-function-state



```
const Circle = ({ cx, cy, r }) ⇒ { … };

const Grid = ({ children, height, width, columns, rows }) ⇒ { … };

const NestedCircles = ({ cx, cy, r }) ⇒ { … };

const Graphic = () ⇒ {
  return (
    <svg width="400" height="400" stroke="black" fill="none">
      <Grid width={400} height={400} columns={2} rows={2}>
        <NestedCircles cx="100" cy="100" numberOfCircles={10} radius={90} />
      </Grid>
    </svg>
  );
};
```



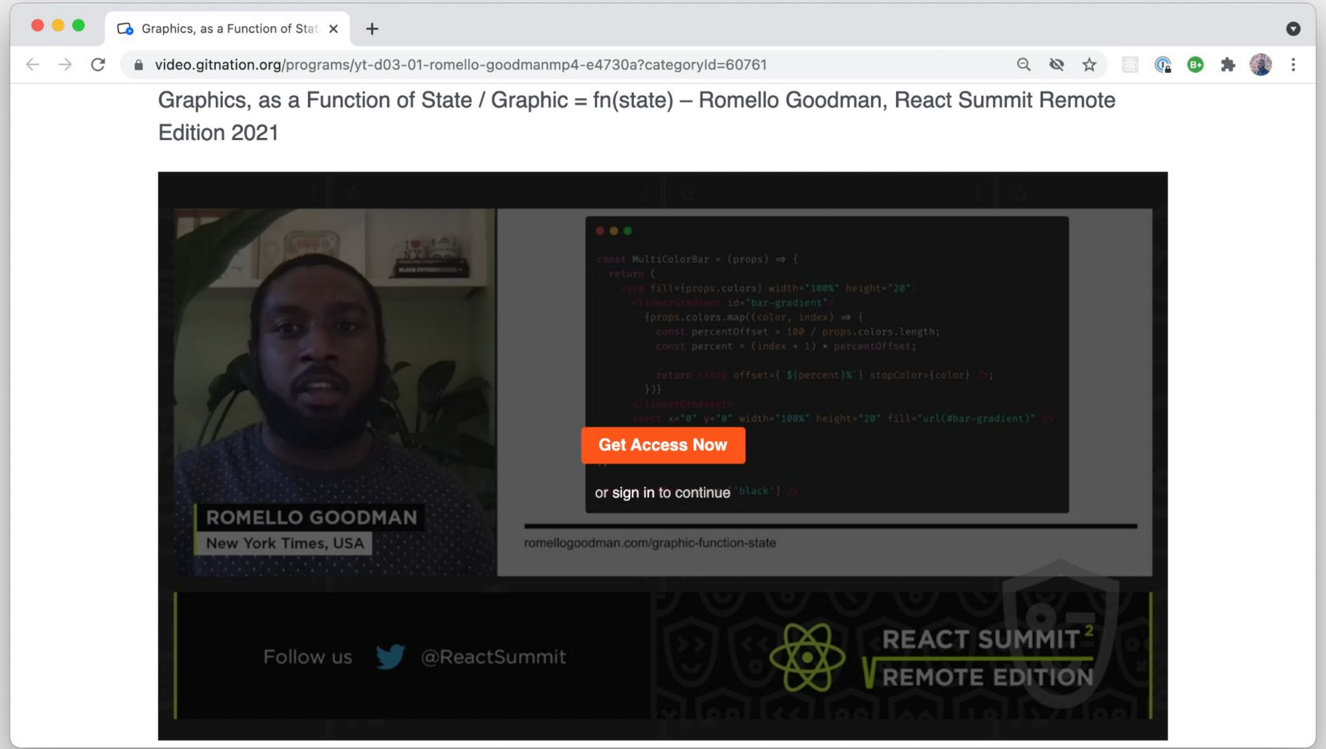
romellogoodman.com/graphic-function-state

Graphic = fn(state)

- In React, your UI is a representation of your state
- Use SVGs to apply React principles to Graphic Design
- State can be contain infinite possibilities
- Graphic Designs can be dynamic and reactive

Thank you for coming 👋🏿👋🏿👋🏿

- 👨🏿‍💻


Graphics, as a Function of Stat
video.gitnation.org/programs/yt-d03-01-romello-goodmanmp4-e4730a?categoryId=60761

Graphics, as a Function of State / Graphic = fn(state) – Romello Goodman, React Summit Remote Edition 2021

```
const MultiColorBar = (props) => {
  return (
    <svg fill={props.colors} width="100%" height="20">
      <linearGradient id="bar-gradient">
        {props.colors.map((color, index) => {
          const percentOffset = 100 / props.colors.length;
          const percent = (index + 1) * percentOffset;

          return <stop offset={`${percent}%`} stopColor={color} />;
        })}
      </linearGradient>
      <rect x="0" y="0" width="100%" height="20" fill="url(#bar-gradient)" />
```

Get Access Now
or sign in to continue
ROMELLO GOODMAN
New York Times, USA
romellogoodman.com/graphic-function-state
Follow us @ReactSummit
REACT SUMMIT
REMOTE EDITION

Watch the recording: Here!

romellogoodman.com/graphic-function-state