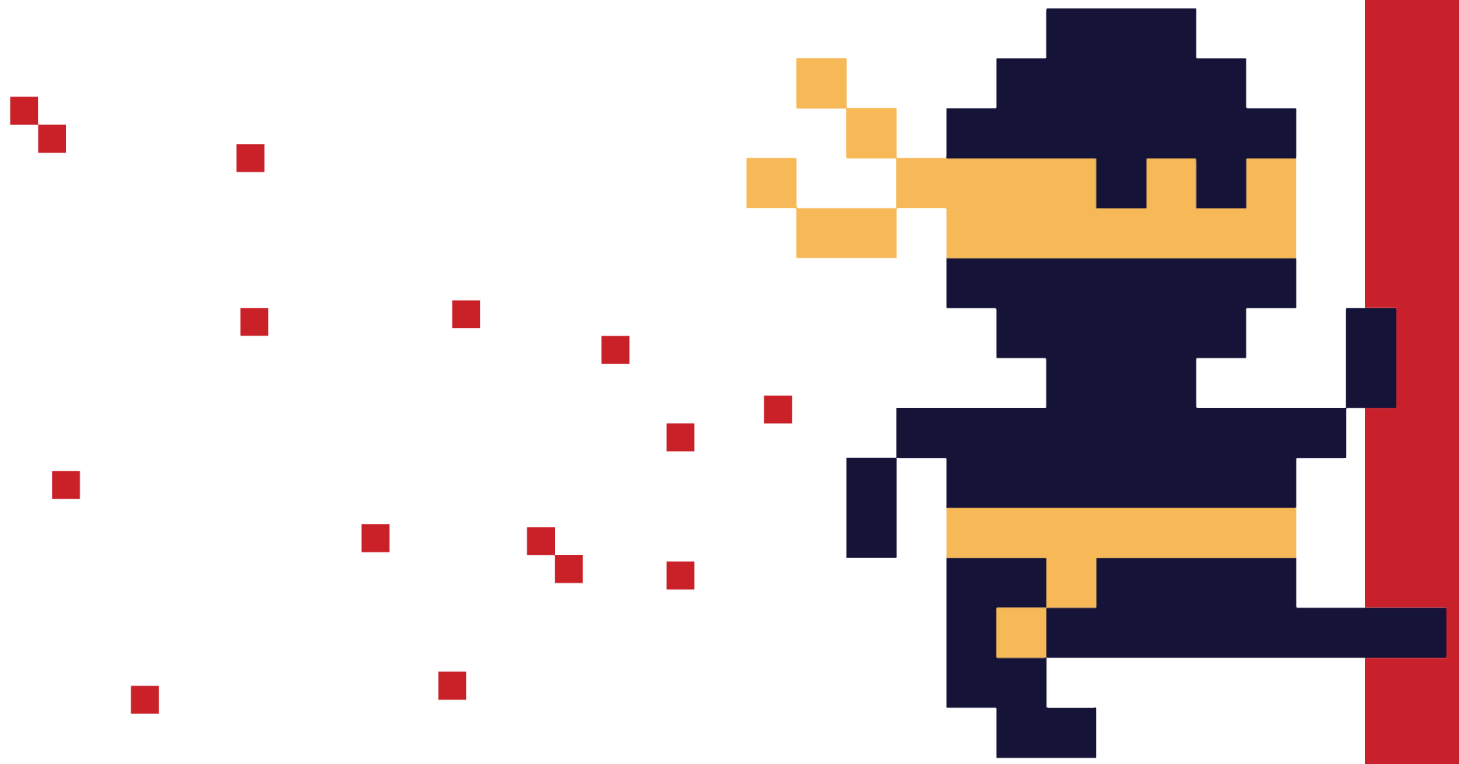
ODESSA JS



Summer. Sea. JavaScript.

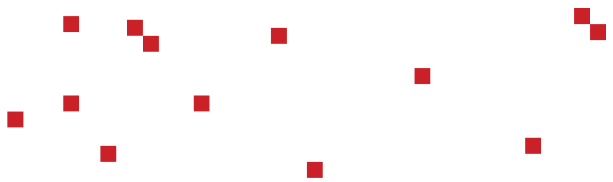
Illia Olenchenko

@icrosil

React + D3 = 💞

React + D3 = ?

- recharts
- nivo
- britecharts react
- vx
- reaviz.io
- victory



React + D3 = ?

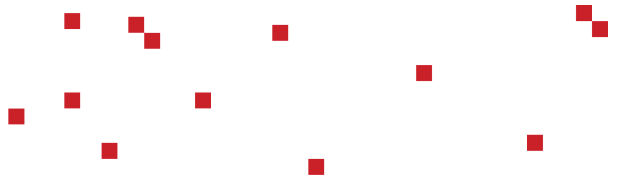
Плюсы

- Консистентность с кодом
- Свобода стиля
- Контроль над компонентами

Минусы

- Разбираться с D3
- Кривая входа в проект хуже
- Поддерживать свое решение

А продакты знают?



С чего начать?

React

- Декларационное отображение
- Управление DOM
- Управление состоянием
- Передача данных

✅ Components
✅ Rendering
✅ State
✅ Props

С чего начать?

D3

- Императивный стиль
- Управление DOM
- Математические функции

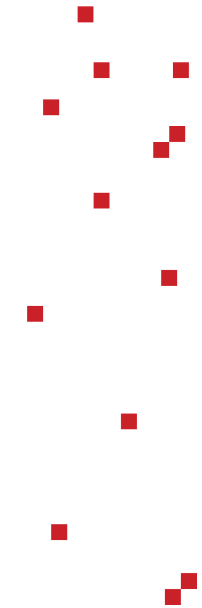
❌ Не сочетается с React
❌ Меняет элементы напрямую в DOM
✅ Умеет просчитывать путь, оси, гео и многое другое.

Хороший/плохой пример

ODESSA JS

renderAxis

```
class RenderAxis extends Component {
  renderAxis() {
    const { hideDomain, textStyle, lineStyle, d3Axis, scale, renderAxis } = this.props;

    const axisElement = d3.select(this.axis).call(d3Axis.scale(scale));

    // remove domain line
    if (hideDomain) axisElement.select('.domain').remove();

    // add styles to textElements
    axisElement
      .selectAll('.tick text')
      .style('fill', textStyle.fill)
      .style('stroke', textStyle.stroke)
      .style('font-size', textStyle.fontSize);

    forEach(lineStyle, (style, key) => {
      axisElement.selectAll('.tick line').attr(key, style);
    });
  }

  // lifecycles
  render() {}
}
```



Хороший/плохой пример

ODESSA JS

renderAxis

```
class RenderAxis extends Component {
  renderAxis() {
    const { hideDomain, textStyle, lineStyle, d3Axis, scale, renderAxis } = this.props;

    const axisElement = d3.select(this.axis).call(d3Axis.scale(scale));

    // remove domain line
    if (hideDomain) axisElement.select('.domain').remove();

    // add styles to textElements
    axisElement
      .selectAll('.tick text')
      .style('fill', textStyle.fill)
      .style('stroke', textStyle.stroke)
      .style('font-size', textStyle.fontSi
```

props

Хороший/плохой пример

ODESSA JS

renderAxis

```
class RenderAxis extends Component {
  renderAxis() {
    const { hideDomain, textStyle, lineStyle, d3Axis, scale, renderAxis } = this.props;

    const axisElement = d3.select(this.axis).call(d3Axis.scale(scale));

    // remove domain line
    if (hideDomain) axisElement.select('.domain').remove();

    // add styles to textElements
    axisElement
      .selectAll('.tick text')
      .style('fill', textStyle.fill)
      .style('stroke', textStyle.stroke)
      .style('font-size', textStyle.fontSize);

    forEach(lineStyle, (style, key) => {
      axisElement.selectAll('.tick line').attr(key, style);
    });
```

axis

SVG vs Canvas

Choose your fighter

SVG vs Canvas

SVG

- ✅ Декларативные компоненты (path/circle)
- ✅ Интерактивность с пользователем
- ❌ Производительность

- 🛠️ Особенности
  - ❌ стилизация
  - ❌ нет z-index
  - ❌ fit text
  - ✅ foreignObject

Canvas

- ❌ Императивное изменение контекста
- ❌ Слабая взаимосвязь с React
- ✅ Производительность

- 🛠️ Особенности
  - ❌ стилизация

SVG vs Canvas

```
const c = document.getElementById("myCanvas");
const ctx = c.getContext("2d");
ctx.beginPath();
ctx.arc(100, 75, 50, 0, 2 * Math.PI);
ctx.stroke();
```

```
<svg height="100" width="100">
  <circle
    cx="50"
    cy="50"
    r="40"
    stroke="black"
    stroke-width="3"
    fill="red"
  />
</svg>
```

“Просто” canvas

ODESSAJS

Canvas

```
function CanvasComponent() {
  const canvasEl = useRef(null);

  useEffect(() => {
    const ctx = canvasEl.current?.getContext('2d');
    ctx.fillRect(0, 0, 100, 100);
  }, [canvasEl]);

  return (
    <canvas ref={canvasEl} width={300} height={300} />
  );
}
```

first render

“Просто” canvas

ODESSAJS

Canvas

```
function CanvasComponent() {
  const canvasEl = useRef(null);

  useEffect(() => {
    const ctx = canvasEl.current?.getContext('2d');
    ctx.fillRect(0, 0, 100, 100);
  }, [canvasEl]);

  return (
    <canvas ref={canvasEl} width={300} height={300} />
  );
}
```

first render

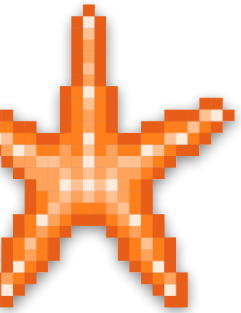
Просто SVG

ODESSA JS

SVG

```
function SVGComponent() {
  return (
    <svg width={300} height={300}>
      <line />
      <path />
      <Axis />
    </svg>
  );
}
```

React way



ODESSA JS

D3 методы

Статистика
- extent (min and max)
- median
- mean
- deviation

Поиск
- bisect

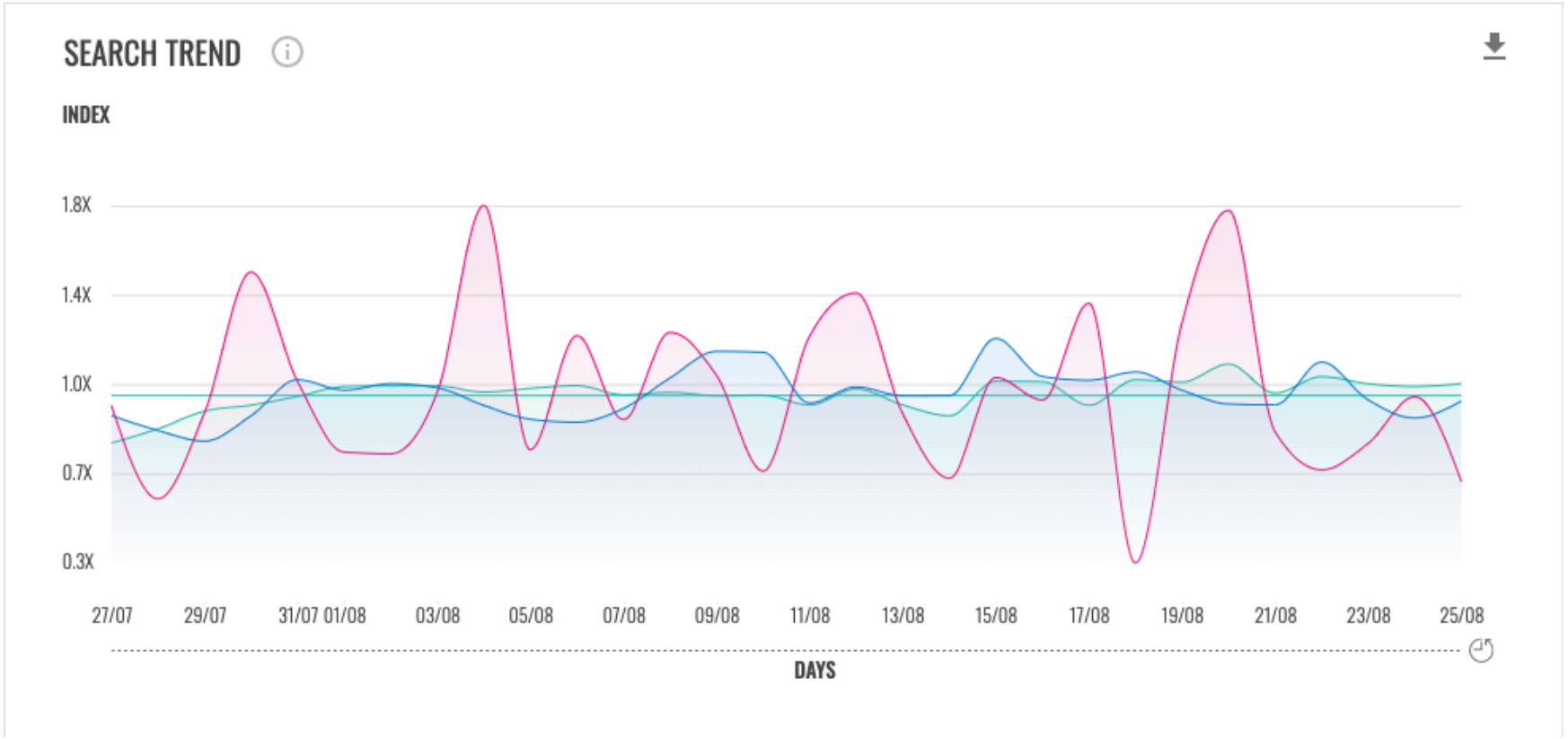
Графики
- histogram
- axes
- brushes
- chord
- nests
- drag
- force
- geo

Цвета
- rgb/hsl/hcl/hex
- darker/brighter

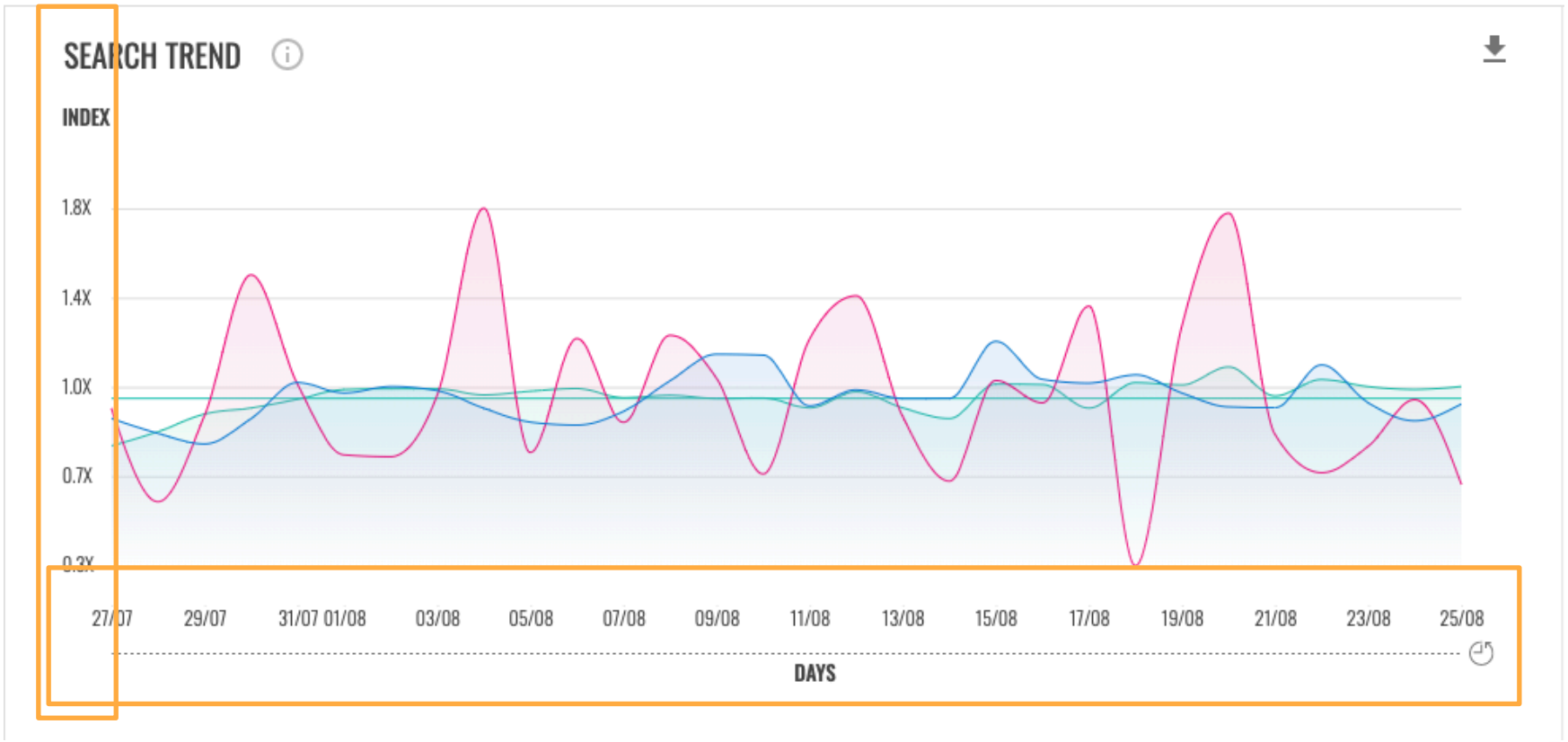
Еще
- contour
- easings (animations)
- formatters
- projections
- hierarchy
- interpolate
- scale
- arcs
- pies
- area
- curve
- zoom

React - view; D3 - math

TimeLine


SEARCH TREND
INDEX
1.8X
1.4X
1.0X
0.7X
0.3X
27/07
29/07
31/07 01/08
03/08
05/08
07/08
09/08
11/08
13/08
15/08
17/08
19/08
21/08
23/08
25/08
DAYS

TimeLine



Ocu

Scales

ODESSA JS

scale

```
export const yScale = (yValue, { height }) => {
  const maxDomain = Math.max(...yValues);
  const minDomain = Math.min(...yValues);

  return scaleLinear()
    .domain([minDomain, maxDomain])
    .range([0, height]);
};

export const xScale = (xValue, { width }) => {
  const maxDomain = Math.max(...xValues);
  const minDomain = Math.min(...xValues);

  return scaleTime()
    .domain([minDomain, maxDomain])
    .range([0, width]);
};
```

Scales

scale

```
export const yScale = (yValue, { height }) => {
  const maxDomain = Math.max(...yValues);
  const minDomain = Math.min(...yValues);

  return scaleLinear()
    .domain([minDomain, maxDomain])
    .range([0, height]);
};

export const xScale = (xValue, { width }) => {
  const maxDomain = Math.max(...xValues);
  const minDomain = Math.min(...xValues);

  return scaleTime()
```

Axis

ODESSA JS

AxisHook

```
function AxisHook({ d3Axis, scale, translateX, translateY, renderAsap }) {
  const ref = useRef(renderAsap ? document.createElement('g') : undefined);

  useEffect(() => {
    d3.select(ref.current).call(d3Axis.scale(scale));
  }, [d3Axis, scale])

  if (renderAsap) {
    ref.current.setAttribute(
      'transform',
      `translate(${translateX}, ${translateY})`,
    );
    d3.select(ref.current).call(d3Axis.scale(scale));
    const [axis] = reactHtmlParser(ref.current.outerHTML);
    return axis;
  }

  return <g ref={ref} transform={`translate(${translateX}, ${translateY})`} />
}
```

Axis

ODESSA JS

AxisHook

```
function AxisHook({ d3Axis, scale, translateX, translateY, renderAsap }) {
  const ref = useRef(renderAsap ? document.createElement('g') : undefined);

  useEffect(() => {
    d3.select(ref.current).call(d3Axis.scale(scale));
  }, [d3Axis, scale])

  if (renderAsap) {
    ref.current.setAttribute(
      'transform',
      `translate(${translateX}, ${translateY})`,
    );
    d3.select(ref.current).call(d3Axis.scale(scale));
    const [axis] = reactHtmlParser(ref.current.outerHTML);
    return axis;
  }

  return <g ref={ref} transform={`translate(${translateX}, ${translateY})`} />
}
```

simple render

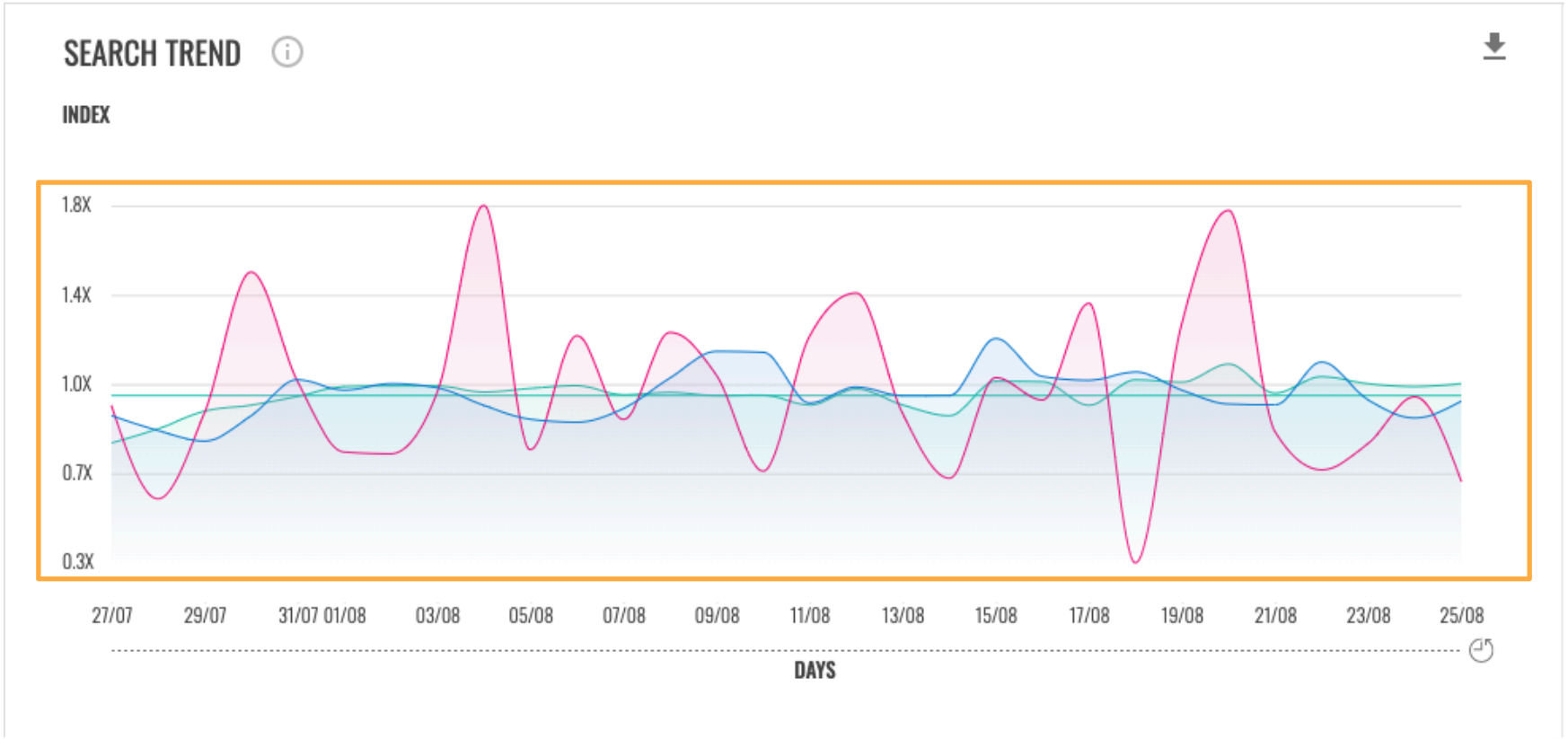
TimeLine component

ODESSA JS

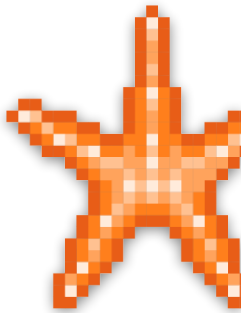
TimeLine

```
function TimeLine({ viewBox, svgProps }) {
  const [data] = useState(defaultData);

  const [xScale, yScale, line, area] = useMemo(
    () => getScales({ viewBox, data }),
    [viewBox, data],
  );

  return (
    <section>
      <svg {...svgProps}>
        <Stripe color={colors.stripe} scale={yScale} />
        {data.map(({ line, id }) => (
          <GradientLine key={id} data={line} />
        ))}
        <AxisX scale={xScale} translateY={dimension.height} />
        <AxisY scale={yScale} />
      </svg>
    </section>
  );
}
```

TimeLine



Сетка + Данные



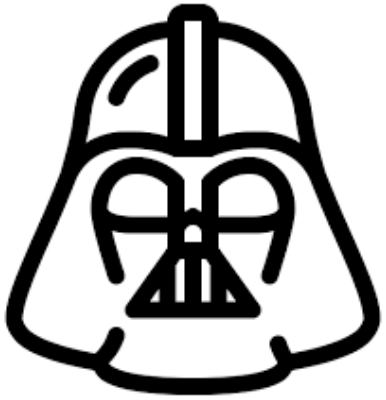
TimeLine component

TimeLine

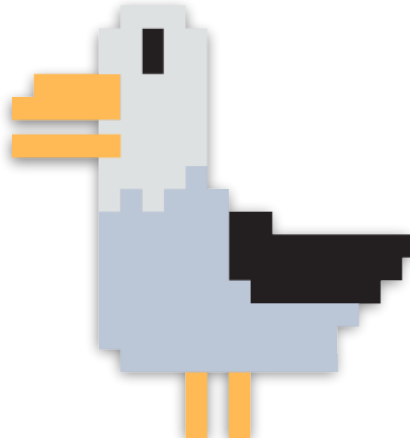
```
function TimeLine({ viewBox, svgProps }) {
  const [data] = useState(defaultData);

  const [xScale, yScale, line, area] = useMemo(
    () => getScales({ viewBox, data }),
    [viewBox, data],
  );

  return (
    <section>
      <svg {...svgProps}>
        <Stripe color={colors.stripe} scale={yScale} />
        {data.map(({ line, id }) => (
          <GradientLine key={id} data={line} />
        ))}
```

scale & data

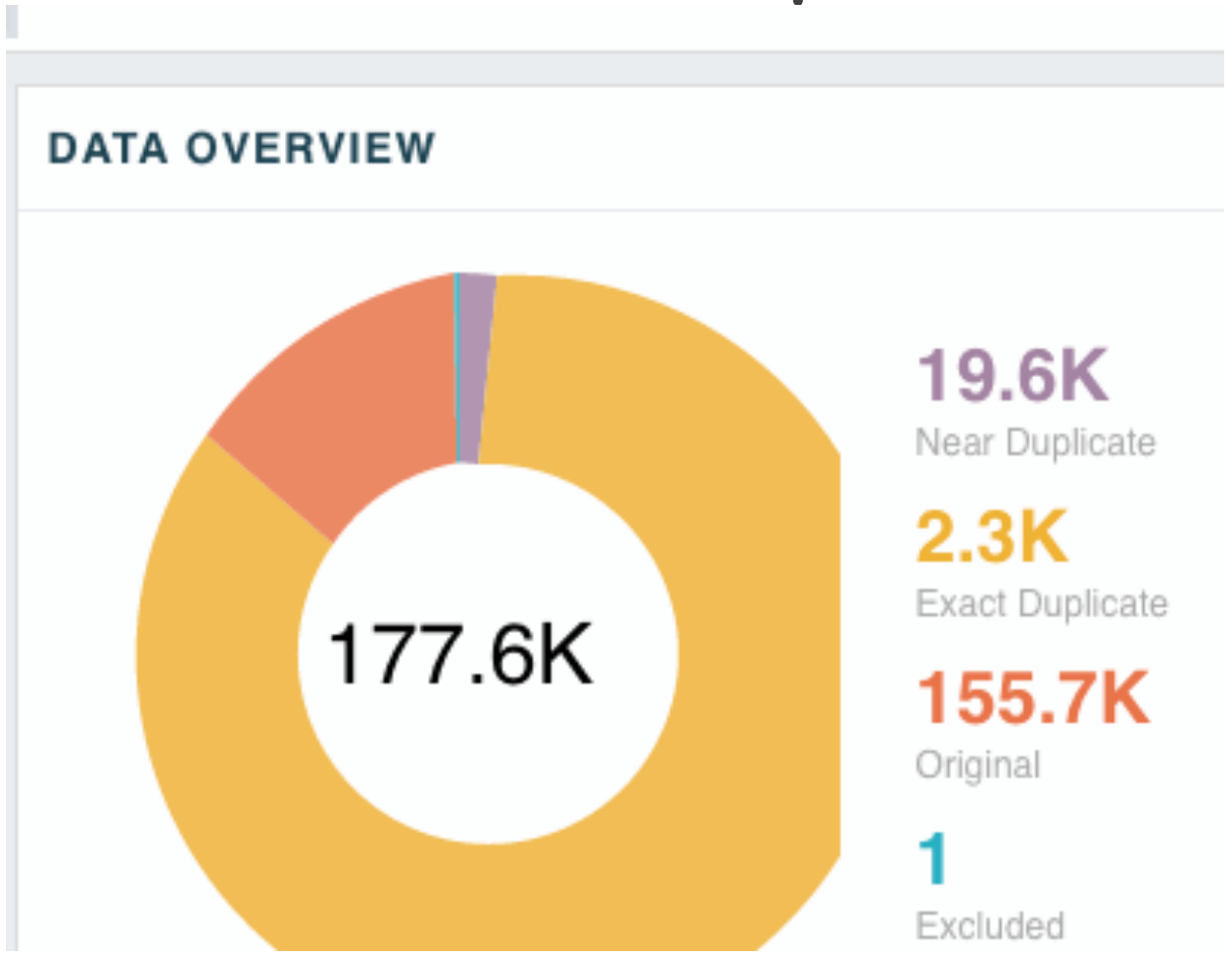
React + D3 пройдено?

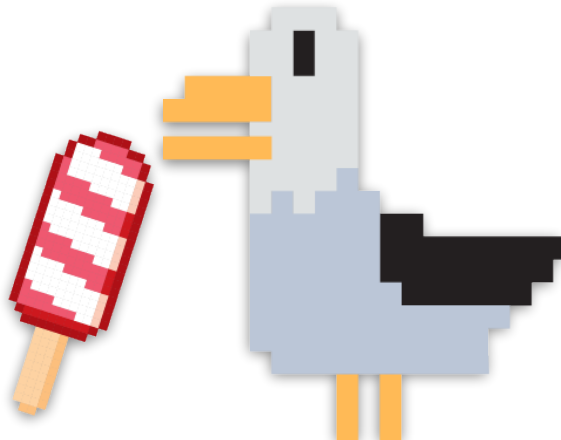


SVG наносит ответный удар

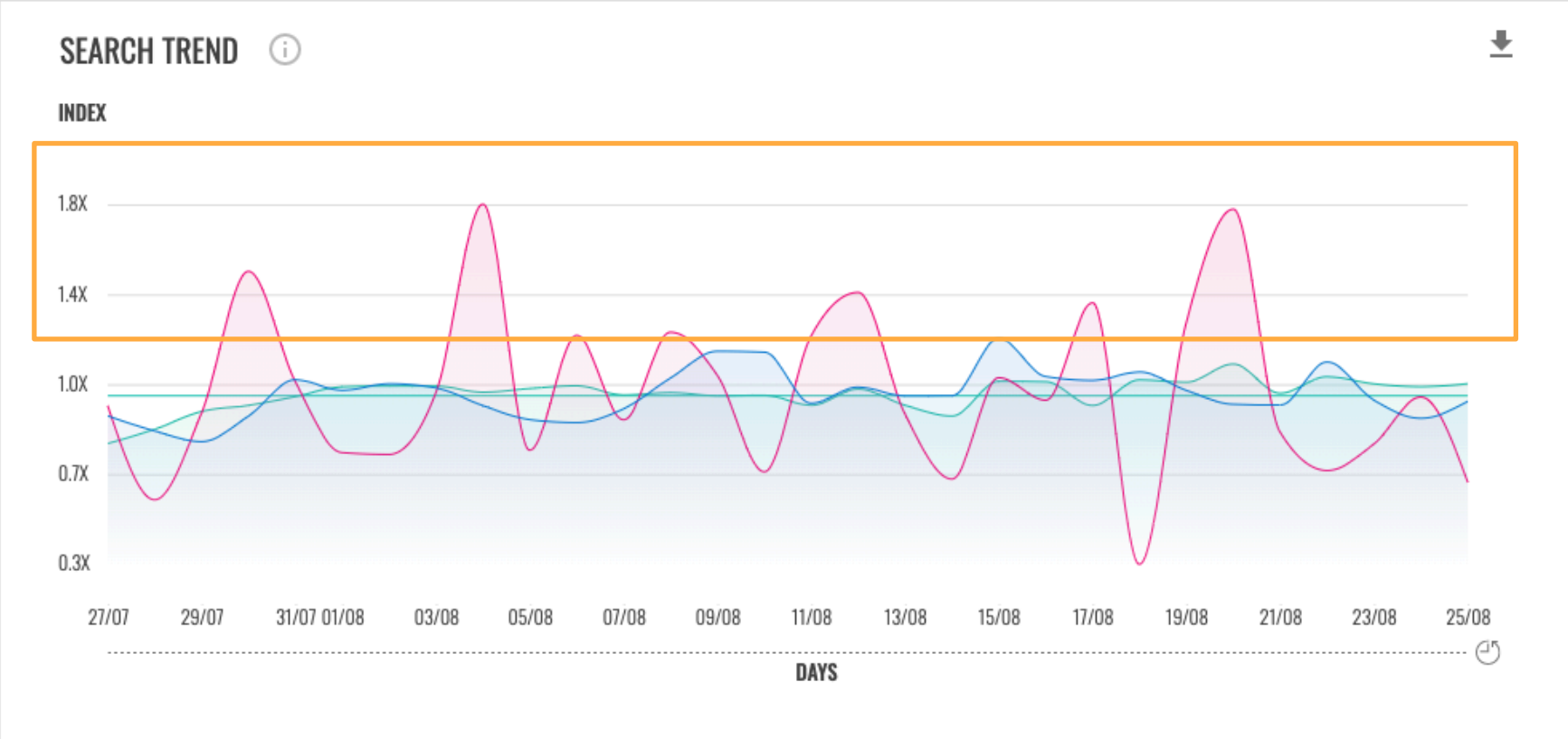


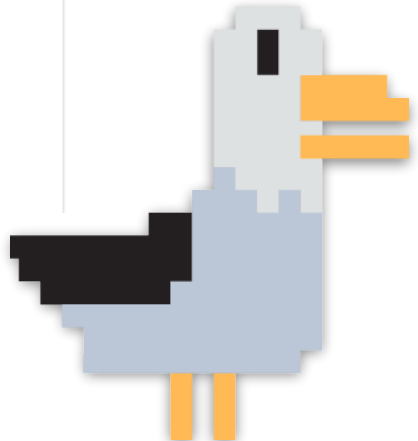
Анимации

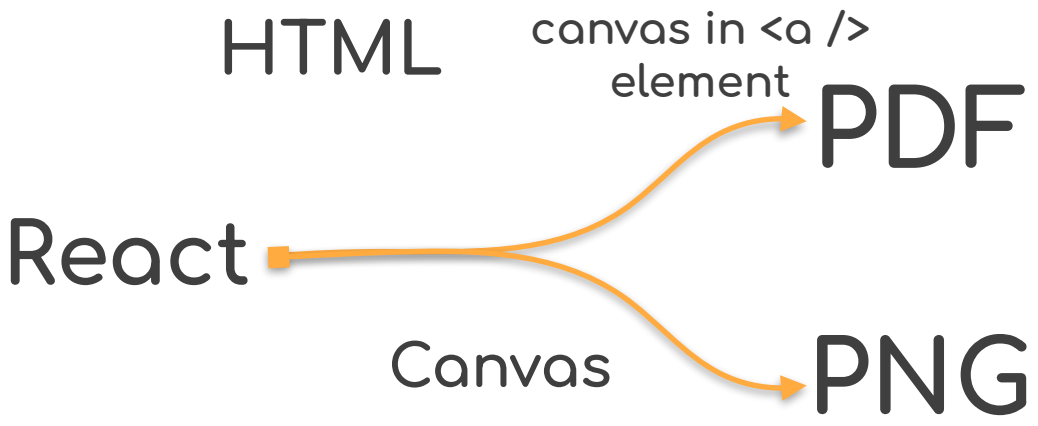

DATA OVERVIEW
177.6K
19.6K
Near Duplicate
2.3K
Exact Duplicate
155.7K
Original
1
Excluded



TimeLine

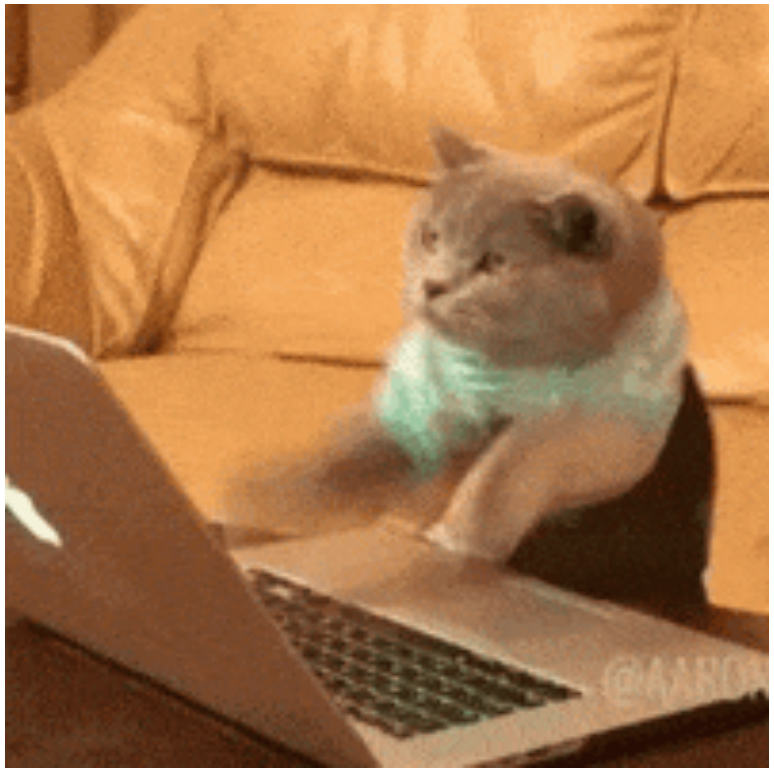

SEARCH TREND
INDEX
1.8X
1.4X
1.0X
0.7X
0.3X
27/07
29/07
31/07 01/08
03/08
05/08
07/08
09/08
11/08
13/08
15/08
17/08
19/08
21/08
23/08
25/08
DAYS




HTML
canvas in <a /> element
PDF
React
Canvas
PNG

Экосистема

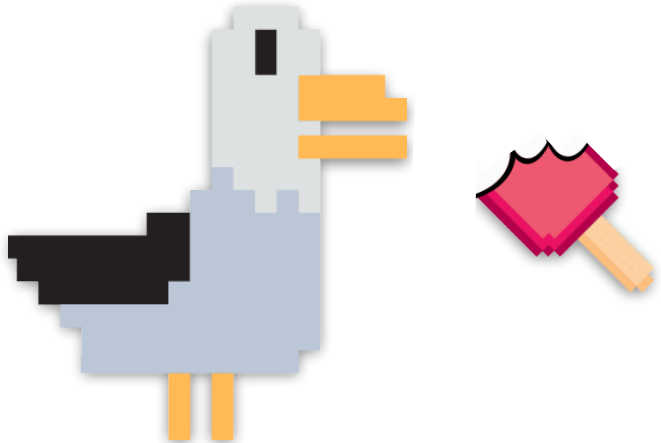
Живой котинг



“Самари”

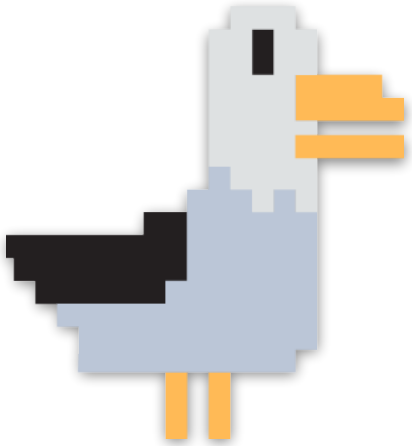
- React - view; D3 - math 😍
- D3 - first подход - только где нужно 🐛
- SVG коварный 📊
- своя библиотека графиков ✅

Спасибо!

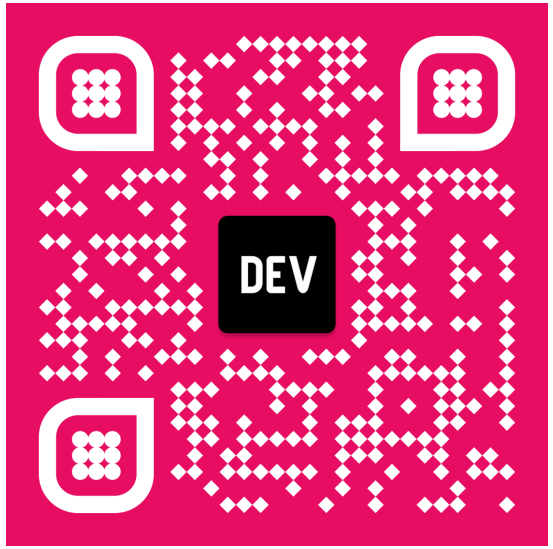


@icrosil

twitter



dev.to



medium