How to use SVG with custom style(color, size)

Read svg files with webpack

To read all of the files in the SVGs folder we're using something special that Webpack gives us, require.context(). This will give us all the relative paths to files in a folder that match a certain regex.

Then we make the **dictionary** object so that we can easily reference the SVGs by key. First we clean up the string so we just have the filename. Then we set the key to be the filename, and have the value be the inlined SVG. To get the SVG inline we'll use svg-inline-loader as a plugin for Webpack. After we **npm install** the package we'll need to add the new loader to the webpack config's **loaders** section (there's more to these sections, but they have been trimmed to focus on the important parts):

assets/icons/index.tsx

```
interface WebpackRequire extends NodeRequire {
    context(file: string, flag?: boolean, exp?: RegExp): any;
}


// this will get fileList of svg
const iconRequireObject = (require as WebpackRequire).context('./svg',
true, /.*\.svg$/); // require all .svgs in the ./svg folder

let dictionary: any = {};

iconRequireObject.keys().forEach((path: string) => {
    const iconName = path.substring(path.lastIndexOf('/') + 1,
path.lastIndexOf('.')); // find the bare name of the icon
    dictionary[iconName] = iconRequireObject(path);
});

export default dictionary;
```

Webpack config

To handle a bunch of SVGs like raw data, we'll need to parse them like raw data. we need A loader for webpack that lets you import files as a string. In your Webpack config, add this in the appropriate place:

webpack.config.js

```
resolve: {
        // Add '.ts' and '.tsx' as resolvable extensions.
        extensions: [".ts", ".tsx", ".js", ".json", ".svg"]
    },
module: {
  rules: [
 .....
    {
      test: /\.svg$/,
      loader: 'svg-inline-loader' // or 'raw-loader'
    }
  ]
}
```

```
yarn add svg-inline-loader --dev
```

Icon Component

IconComponent.tsx

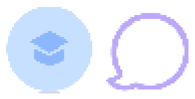
```
import React from 'react';
import iconSVGData from '../common/assets/icons';

interface IConProps {
    name: string,
    color?: string,
    className? : string,
    size? : number,
}

const IconComponent = (props: IConProps) => (
    <svg
        style={{ fill: props.color }}
        className={ props.className }
        width={ props.size }
        height={ props.size }
        dangerouslySetInnerHTML={ { __html: iconSVGData[props.name] } }
    >
    </svg>
);


export default IconComponent;
```

What we get svg file from designer



learn.svg

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Generator: Adobe Illustrator 19.2.1, SVG Export Plug-In . SVG
Version: 6.00 Build 0)  -->
<svg version="1.1" id="_1" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink" x="0px" y="0px"
   viewBox="0 0 50 50" style="enable-background:new 0 0 50 50;"
xml:space="preserve">
<style type="text/css">
  .st0{clip-path:url(#SVGID_2_);}
  .st1{clip-path:url(#SVGID_6_);}
  .st2{opacity:0.5;clip-path:url(#SVGID_8_);fill:#C5DCFF;}
  .st3{clip-path:url(#SVGID_10_);}
  .st4{clip-path:url(#SVGID_12_);fill:#87B5FF;}
</style>
<g>
  <defs>
    <polygon id="SVGID_1_" points="27.1,13.2 27.1,25.1 34.2,25.1 22.1,40
22.1,29.1 15,29.1    "/>
  </defs>
  <clipPath id="SVGID_2_">
    <use xlink:href="#SVGID_1_"  style="overflow:visible;"/>
  </clipPath>
  <g class="st0">
    <defs>
      <rect id="SVGID_3_" width="50" height="50"/>
    </defs>
    <clipPath id="SVGID_4_">
      <use xlink:href="#SVGID_3_"  style="overflow:visible;"/>
    </clipPath>
  </g>
</g>
<g>
  <defs>
    <circle id="SVGID_5_" cx="25" cy="25" r="25"/>
  </defs>
  <clipPath id="SVGID_6_">
    <use xlink:href="#SVGID_5_"  style="overflow:visible;"/>
  </clipPath>
  <g class="st1">
    <defs>
      <rect id="SVGID_7_" width="50" height="50"/>
    </defs>
    <clipPath id="SVGID_8_">
      <use xlink:href="#SVGID_7_"  style="overflow:visible;"/>
    </clipPath>
    <rect x="-5" y="-5" class="st2" width="60" height="60"/>
  </g>
</g>
<g>
  <defs>
    <path id="SVGID_9_"
d="M33.2,25l-7.9,4c-0.2,0.1-0.4,0.1-0.5,0l-7.9-4c-0.4-0.2-0.8,0.1-0.8,0.
5v5.1c0,0.2,0.1,0.4,0.3,0.5

c0.9,0.7,4.8,3.5,8.7,3.5c3.9,0,7.8-2.7,8.7-3.5C34,31,34,30.8,34,30.6v-5.
2C34,25,33.6,24.8,33.2,25z M35.7,20.3l-10.4-5.7

c-0.2-0.1-0.4-0.1-0.5,0l-10.5,5.7c-0.4,0.2-0.4,0.8,0,1l10.4,5.8c0.2,0.1,
0.4,0.1,0.5,0l10.5-5.7C36.2,21.1,36.2,20.6,35.7,20.3z
      "/>
  </defs>
```

```
<clipPath id="SVGID_10_">
  <use xlink:href="#SVGID_9_"  style="overflow:visible;"/>
</clipPath>
<g class="st3">
  <defs>
    <rect id="SVGID_11_" width="50" height="50"/>
  </defs>
  <clipPath id="SVGID_12_">
    <use xlink:href="#SVGID_11_"  style="overflow:visible;"/>
  </clipPath>
  <rect x="9" y="9.6" class="st4" width="32.1" height="30"/>
  </g>
</g>
</svg>
```

=> we need to get rid of all of other unnecessary tags like this

learn.svg

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Generator: Adobe Illustrator 19.2.1, SVG Export Plug-In . SVG
Version: 6.00 Build 0)  -->
<svg version="1.1" id="_1" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink" x="0px" y="0px"
   viewBox="0 0 50 50" style="enable-background:new 0 0 50 50;"
xml:space="preserve">
  <path
d="M33.2,25l-7.9,4c-0.2,0.1-0.4,0.1-0.5,0l-7.9-4c-0.4-0.2-0.8,0.1-0.8,0.
5v5.1c0,0.2,0.1,0.4,0.3,0.5

c0.9,0.7,4.8,3.5,8.7,3.5c3.9,0,7.8-2.7,8.7-3.5C34,31,34,30.8,34,30.6v-5.
2C34,25,33.6,24.8,33.2,25z M35.7,20.3l-10.4-5.7

c-0.2-0.1-0.4-0.1-0.5,0l-10.5,5.7c-0.4,0.2-0.4,0.8,0,1l10.4,5.8c0.2,0.1,
0.4,0.1,0.5,0l10.5-5.7C36.2,21.1,36.2,20.6,35.7,20.3z
    "/>
</svg>
```

svg-inline-loader options svg Imago svg

webpack.config.js

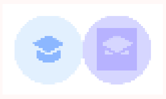
```
{
  test: /\.svg$/,
  loader:"svg-inline-loader",
  options: {
   removeTags: true,
   removingTags: ['title', 'desc', 'style', 'rect', 'clipPath',
'circle', 'polygon'],
 }
}
```

.. <path> <defs> svg-inline-loader child <defs>

auto-generated learn.svg with unneeded crusts

```
<svg version="1.1" id="_1" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink" x="0px" y="0px" viewBox="0 0
50 50" style="enable-background:new 0 0 50 50;" xml:space="preserve">
...
<g>
  <defs>
    <path id="SVGID_9_"
d="M33.2,25l-7.9,4c-0.2,0.1-0.4,0.1-0.5,0l-7.9-4c-0.4-0.2-0.8,0.1-0.8,0.
5v5.1c0,0.2,0.1,0.4,0.3,0.5
c0.9,0.7,4.8,3.5,8.7,3.5c3.9,0,7.8-2.7,8.7-3.5C34,31,34,30.8,34,30.6v-5.
2C34,25,33.6,24.8,33.2,25z M35.7,20.3l-10.4-5.7
c-0.2-0.1-0.4-0.1-0.5,0l-10.5,5.7c-0.4,0.2-0.4,0.8,0,1l10.4,5.8c0.2,0.1,
0.4,0.1,0.5,0l10.5-5.7C36.2,21.1,36.2,20.6,35.7,20.3z ">
    </path>
  </defs>
  <g class="st3"><defs></defs></g>
</g>
</svg>
```

.svg . .svg SVGID_ . .

 =>

svg . color ***fill="none"*** => path ***fill="#87B5FF"***

learn.svg from sketch

```
<?xml version="1.0" encoding="UTF-8"?>
<svg width="50px" height="50px" viewBox="0 0 50 50" version="1.1"
xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink">
    <!-- Generator: Sketch 43.2 (39069) -
http://www.bohemiancoding.com/sketch -->
    <title>Group</title>
    <desc>Created with Sketch.</desc>
    <defs></defs>
    <g id="Symbols" stroke="none" stroke-width="1" fill="none"
fill-rule="evenodd">
        <g id="learn_icon">
            <g id="Group">
                <circle id="Oval" fill="#C5DCFF" opacity="0.5" cx="25"
cy="25" r="25"></circle>
                <path d="M35.7468537,20.2911201 L25.35583,14.6225487
C25.1766744,14.5311201 24.9975188,14.5311201 24.8183632,14.6225487
L14.3377617,20.2911201 C13.8898728,20.4739773 13.8898728,21.1139773
14.3377617,21.2968344 L24.7287854,27.0568344 C24.907941,27.148263
25.0870966,27.148263 25.2662522,27.0568344 L35.7468537,21.388263
C36.1947426,21.1139773 36.1947426,20.5654058 35.7468537,20.2911201 Z
M33.2386755,24.9539773 L25.35583,28.9768344 C25.1766744,29.068263
24.9975188,29.068263 24.8183632,28.9768344 L16.9355176,24.9539773
C16.5772065,24.7711201 16.1293175,25.0454058 16.1293175,25.5025487
L16.1293175,30.6225487 C16.1293175,30.8054058 16.2188953,30.988263
16.3980509,31.0796916 C17.2938288,31.8111201 21.2352516,34.5539773
25.0870966,34.5539773 C28.9389416,34.5539773 32.8803644,31.8111201
33.7761423,31.0796916 C33.9552979,30.988263 34.0448757,30.8054058
34.0448757,30.6225487 L34.0448757,25.4111201 C34.0448757,25.0454058
33.5969867,24.7711201 33.2386755,24.9539773 Z" id="Shape"
fill="#87B5FF"></path>
            </g>
        </g>
    </g>
</svg>
```

learn.svg from zeplin

```
<svg xmlns="http://www.w3.org/2000/svg" width="50" height="50"
viewBox="0 0 50 50">
    <g fill="none" fill-rule="evenodd">
        <circle cx="25" cy="25" r="25" fill="#D8E7FF"/>
        <path fill="#79ADFF" d="M35.747 20.291l-10.391-5.668a.56.56 0 0
0-.538 0l-10.48 5.668a.544.544 0 0 0 0 1.006l10.39 5.76a.56.56 0 0 0
.538 0l10.48-5.669c.449-.274.449-.823 0-1.097zm-2.508 4.663l-7.883
4.023a.56.56 0 0 1-.538
0l-7.882-4.023c-.359-.183-.807.091-.807.549v5.12c0
.182.09.365.27.457.895.731 4.836 3.474 8.688 3.474 3.852 0 7.793-2.743
8.69-3.474a.504.504 0 0 0 .268-.457V25.41c0-.366-.448-.64-.806-.457z"/>
    </g>
</svg>
```

When we use Icon Component with custom Style(inline style and css class)

IconComponent.tsx

```
import React from 'react';
...
import Icon from '../components/IconComponent';

...
render() {
...
 const size = 80;
    const borderRadius = size / 2;
 return (
  {/* inline style */}
     <div style={{ borderRadius: borderRadius, backgroundColor:
'#c5ddff', width: size, height: size }}>
            <Icon name="learn" color='#87b5ff' size={size} />
        </div>
  {/* css style with Class */}
        <div className="icon-color-red-container" style={{ borderRadius:
borderRadius, width: size, height: size }}>
         <Icon className="icon-color-red" name="learn" size={size} />
        </div>
        <Icon name="messages" color='#bdabf5' size={size} />

 );
}
```

CSS style

main.css

```
.icon-color-red-container {
  background-color: #f9c9de;
}

.icon-color-red {
  fill: #f62c73;
}
```

TODOs

with React Native

: convert svg to ttf(vector icons)

- With Fontello/Icomoon and react-native-vector-icons

- Custom React Native Icons? YESSSSS

References

- An odyssey to find a sustainable icon system with SVGs in React
- Managing inline SVGs with React/Webpack