Hello

I'm @DavidWells

FullStack JavaScript Developer

@ SERVERLESS

previously UX/UI

@ MuleSoft™

slides:

bit.ly/////////

The Quest

Shared

Organized

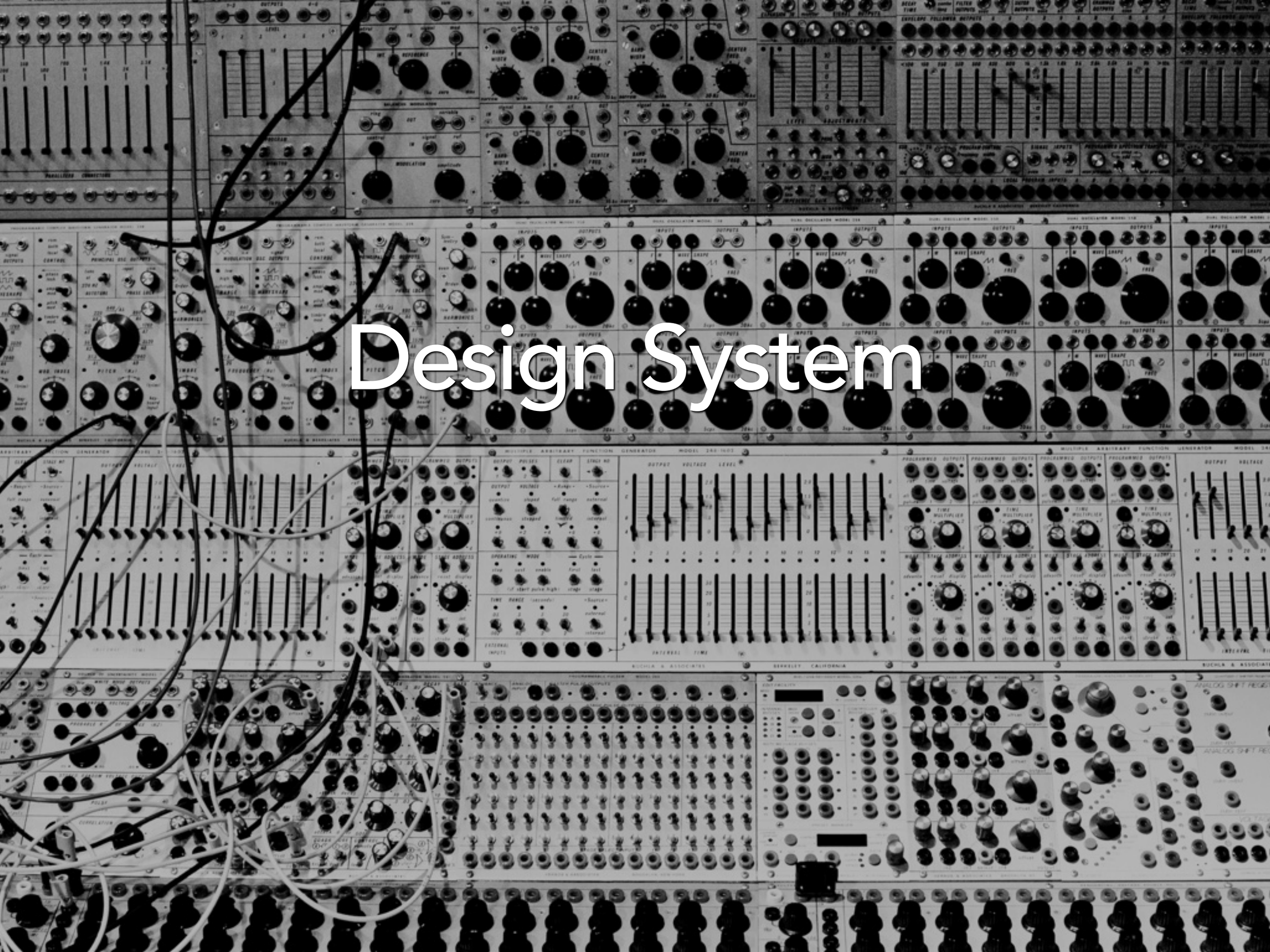
in sync

self contained

Scalable

Platform UI

Component based architectures



Design System

LightningDesignSystem.com

salesforce Design System

Search

- Getting Started
- Platforms
- Guidelines
- Components
- Design Tokens
- Icons
- Downloads
- Articles
- FAQ
- Feedback

Lightning Design System

Create the world's best enterprise app experiences.

Get the Design System | Star

Current release: 2.1.2 | Archives

http://bit.ly/19-oss-libs

-react-component-libraries-to-use-in-your-next-project/

19 Open Source React Component Libraries to use in your next project

Building React components from scratch is a ton of fun.

Tackling the reimplementation of common UI elements gives you the power to bring a ton of UX improvements.

But, sometimes **you just need to get shit done** and recreating the wheel isn't in scope.

So, I've compiled a list of React Component UI libraries to jump start your UI.

These are can be combined, mixed and matched to create your UI pretty quickly.

I've also used many of these libraries as a reference point when implementing my own components.

> How would one create an XYZ component with React?
> What prop types might I need?
> Are their npm dom utilities I can leverage for this?"
> – me. all day er day

ux.mulesoft.com

MuleSoft Components 1.16.2

Components UX patterns Typography Colors Icons Demos Docs

Components

- ActionBar
- ActionButton
- BarChart
- Button
- Card
- Checkbox
- Column
- ContentEditable
- Detail
- DonutChart
- DropdownMenu
- DropdownToggle
- EmptyResult
- FileUploader
- Filter
- FilterBar
- Flex
- Form
- GridLayout
- Heading
- Hero
- HoverContent
- Icon
- InfoMetric

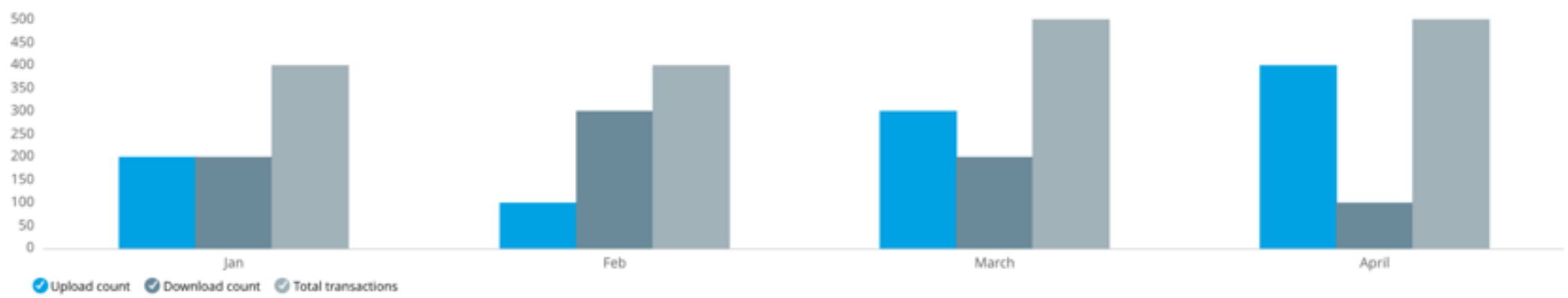
ActionBar

Share Edit Pin Download

ActionButton

BarChart

BarChart is a WIP. Not stable for use.

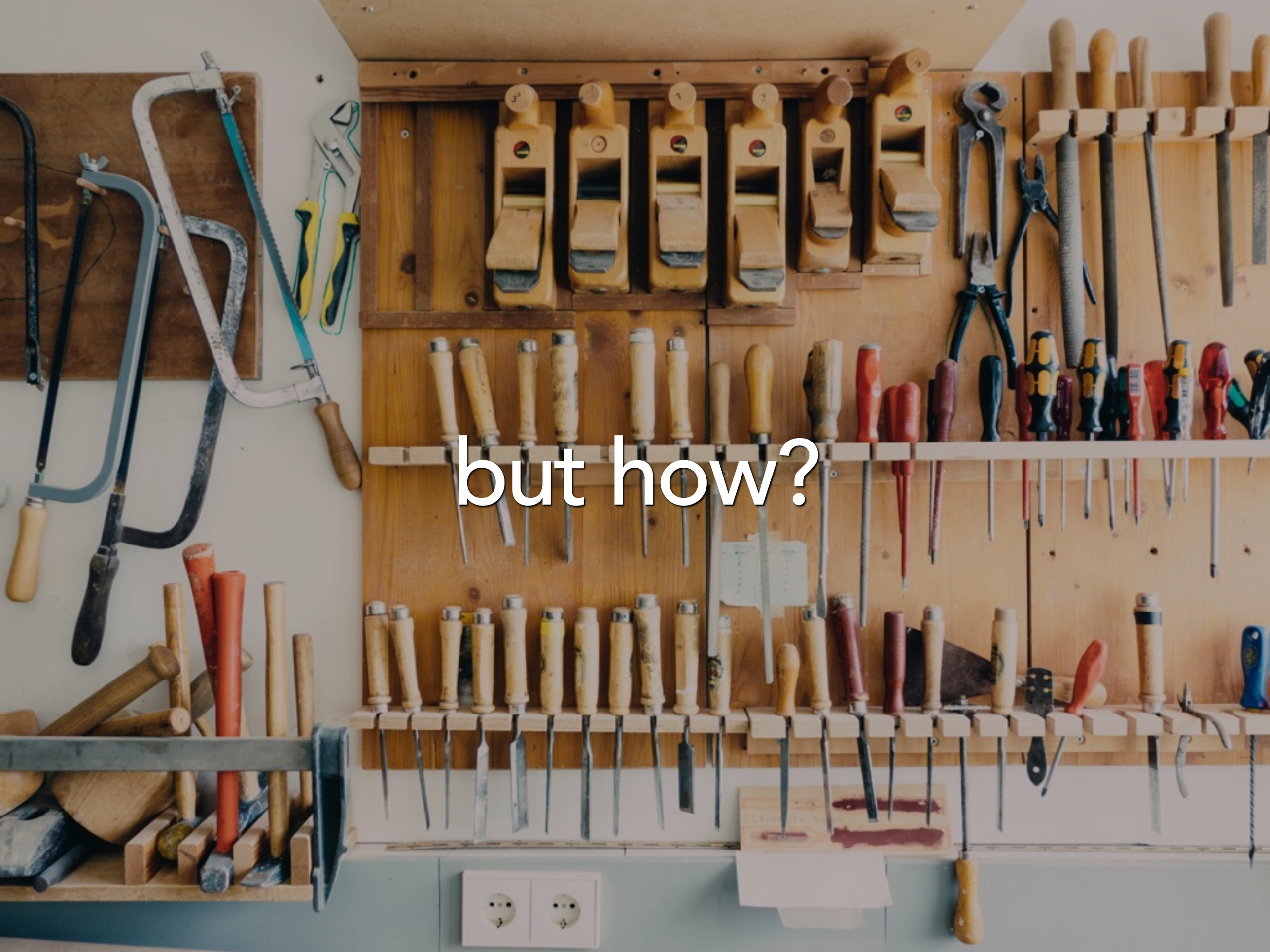


Button

This component generates a button element. It should be used for primary and secondary type actions in the UI

What lives in the style guide?

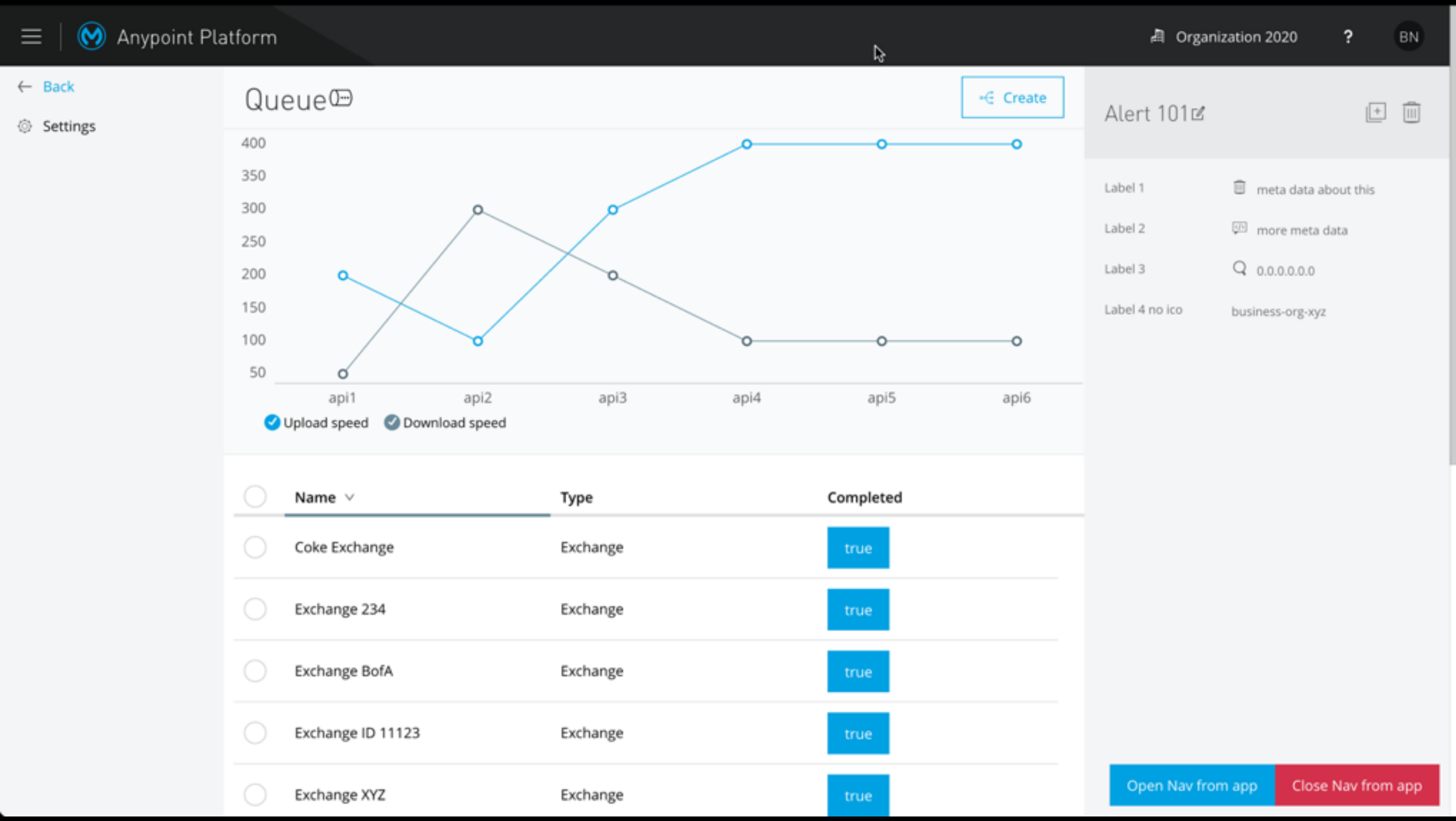
- Documentation
- Live code playground
- Usage examples
- Platform guidelines


but how?

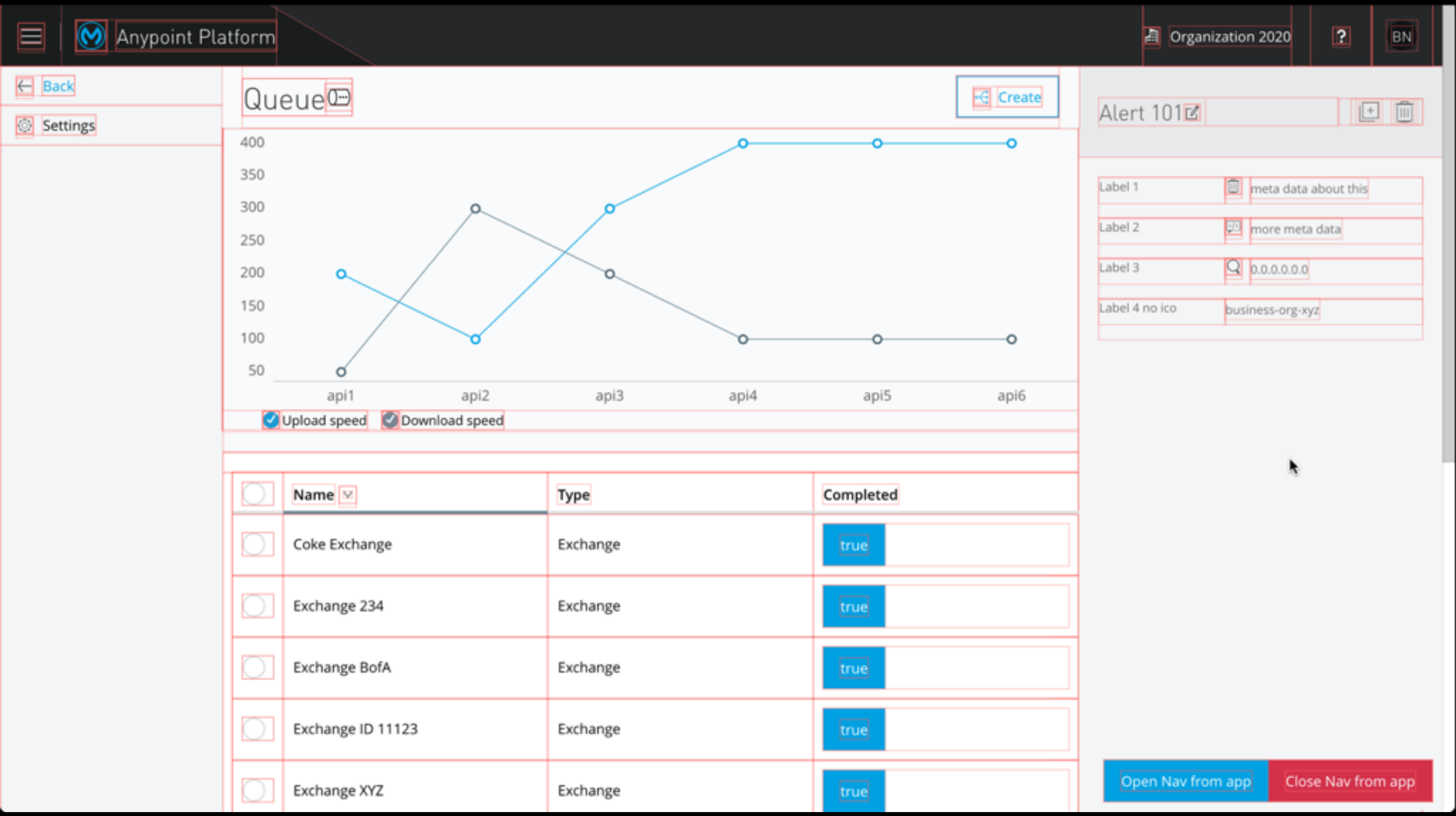
How to build a component lib

1. Breaking things down
2. How to handle CSS?
3. How to global assets like variables, mixing, and icons?
4. How to package them for consumption?
5. profit 🔥

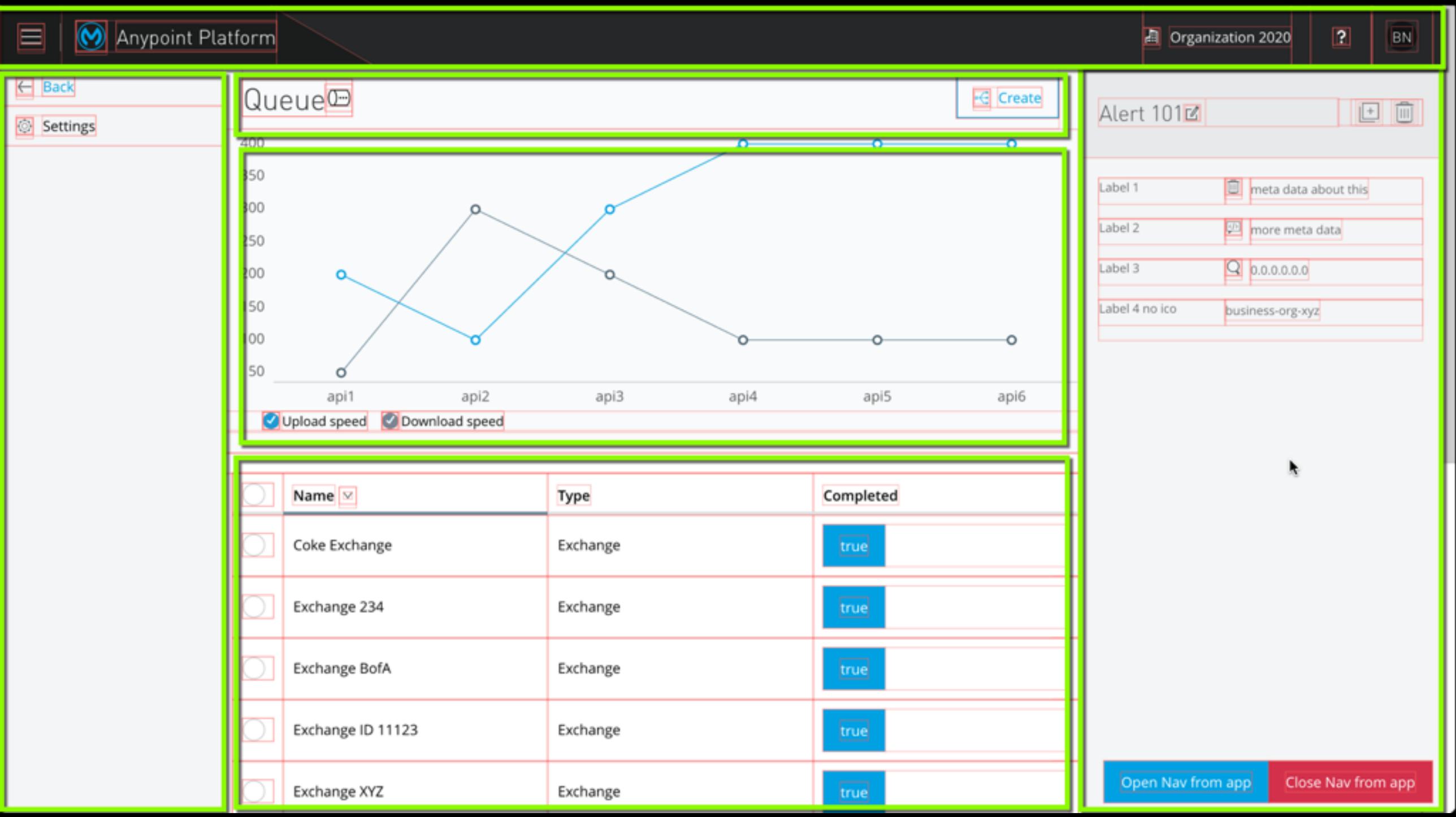
Start by breaking things down



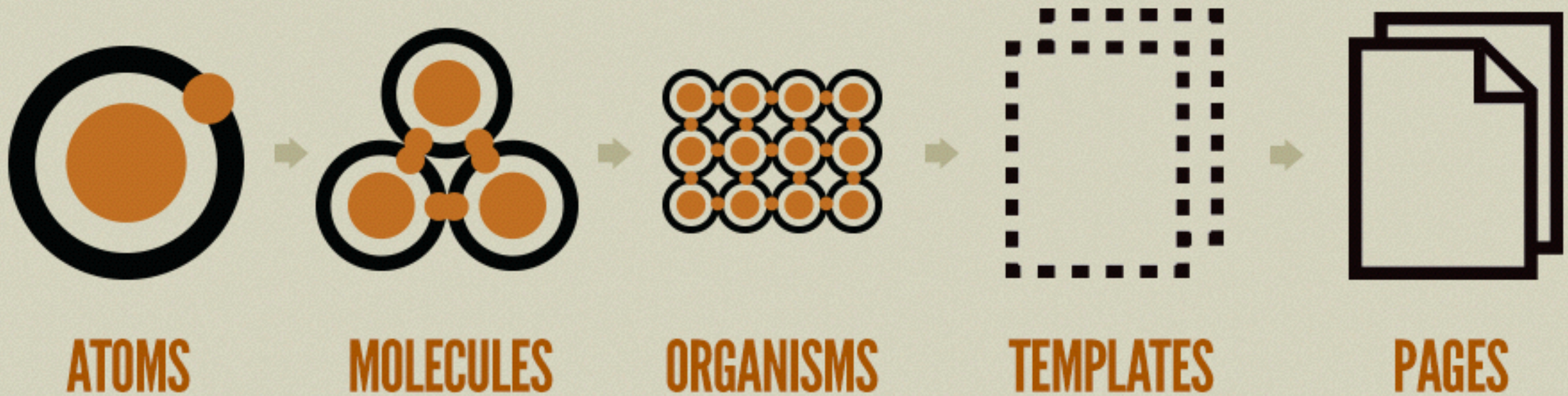
Everything is a component



Smaller components compose into larger ones



Atomic Design



bit.ly/at-design

SEARCH THE SITE

LABEL

ENTER KEYWORD

INPUT

SEARCH

BUTTON

SEARCH THE SITE

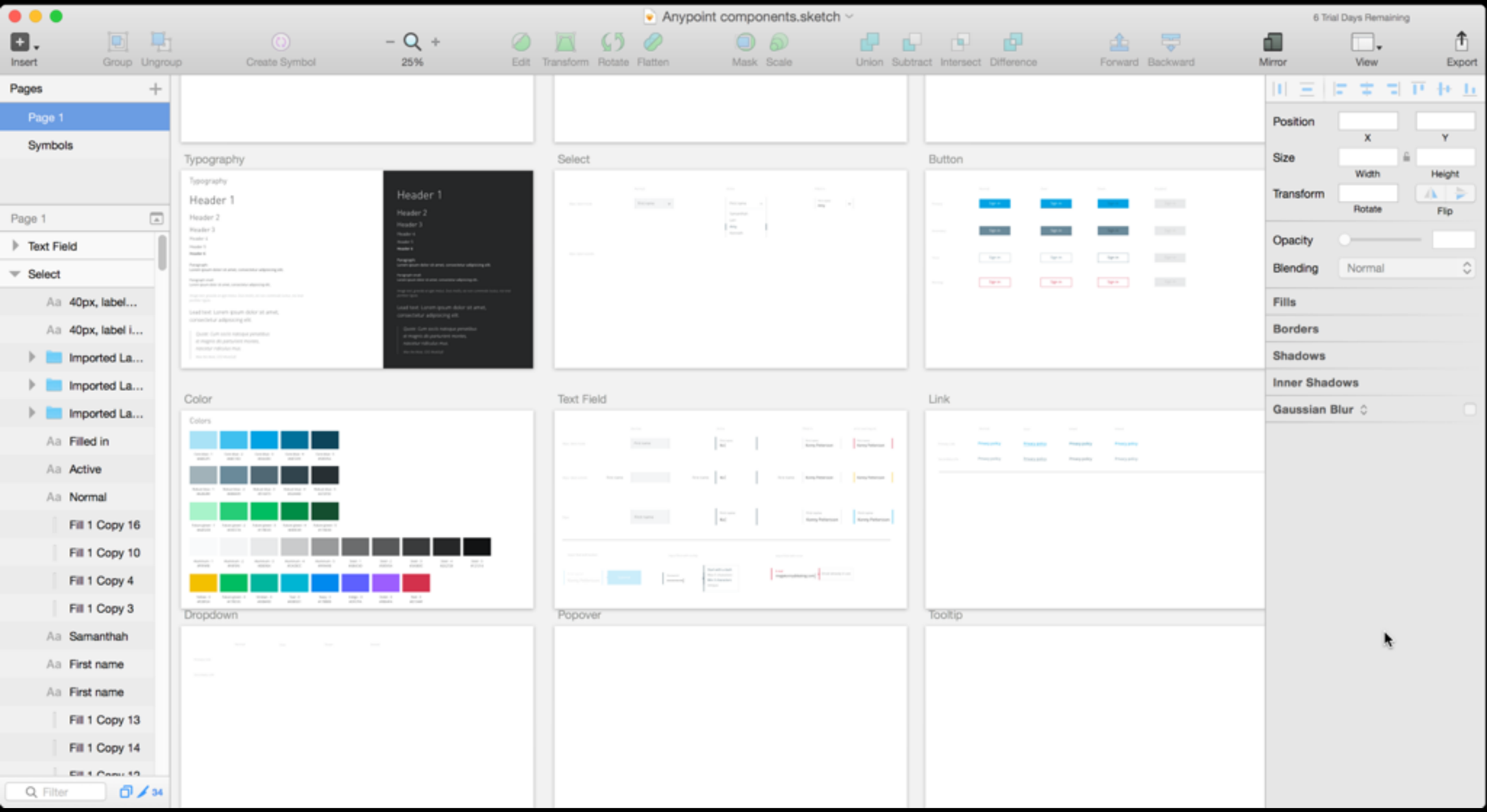
ENTER KEYWORD

SEARCH

Anatomy of a component

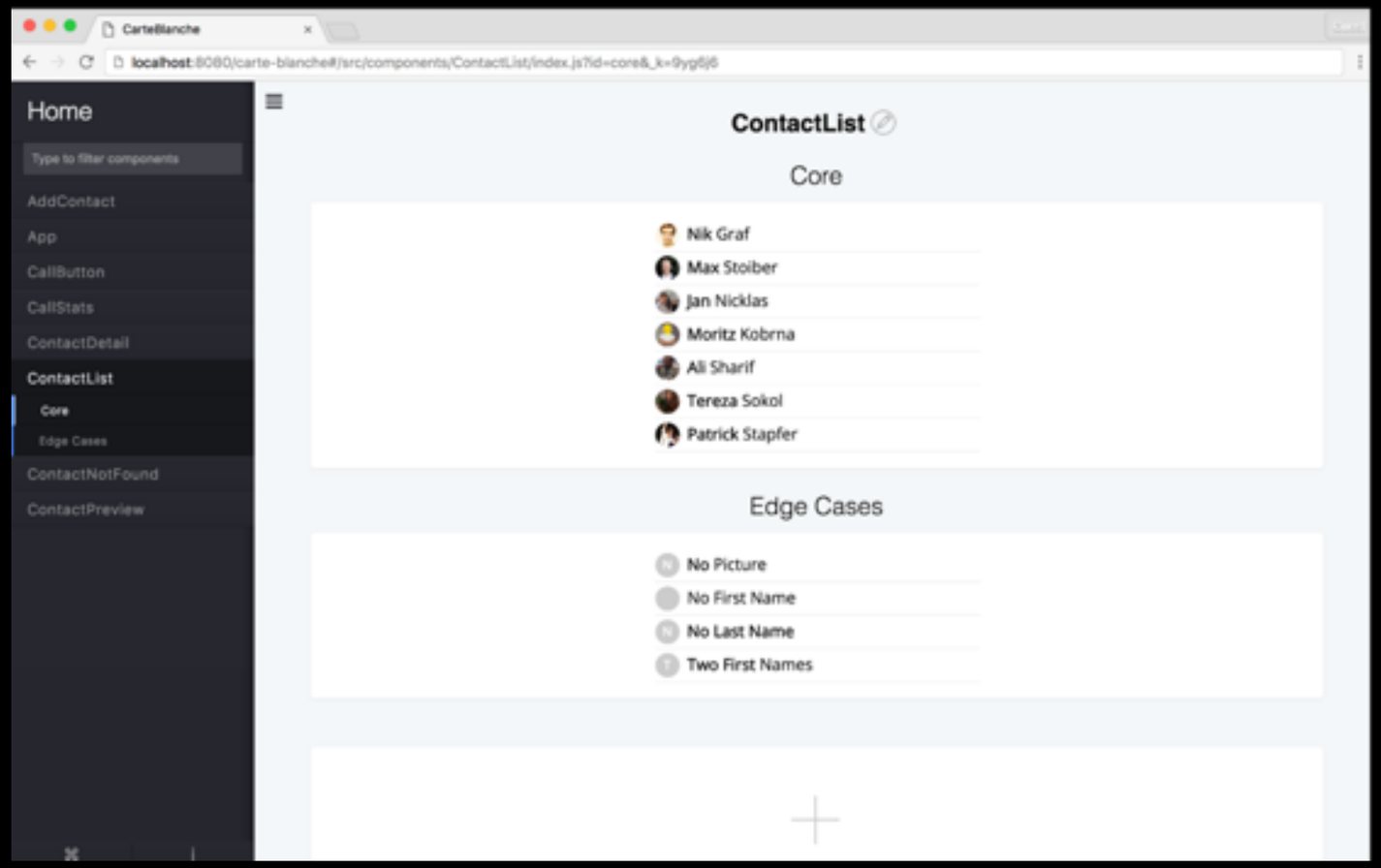
```
/src/components/
  └── ComponentName/            # Component folder
      ├── ComponentName.js        # src code
      ├── ComponentName.css       # styles
      ├── ComponentName.spec.js   # tests
      ├── index.js                # Exports your component
      └── examples/               # Usage Examples
        └── component_usage.js  # Examples of component usage
```

Aside: Working with designers



Tools to help

- Carte-blanche
- React-storybook
- react doc gen

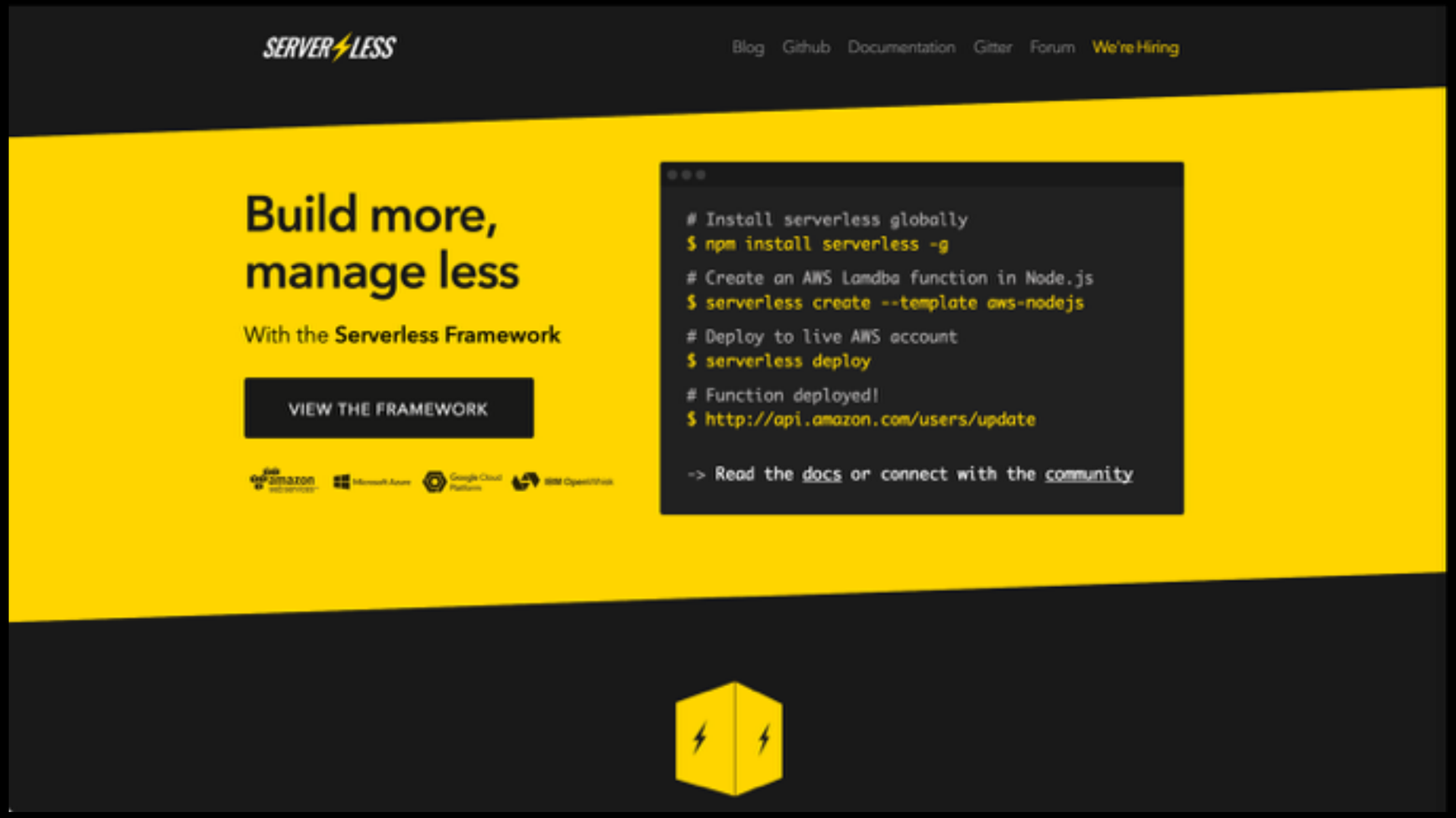


github.com/blai/react-component-illustrator

github.com/blai/react-component-illustrator-webpack

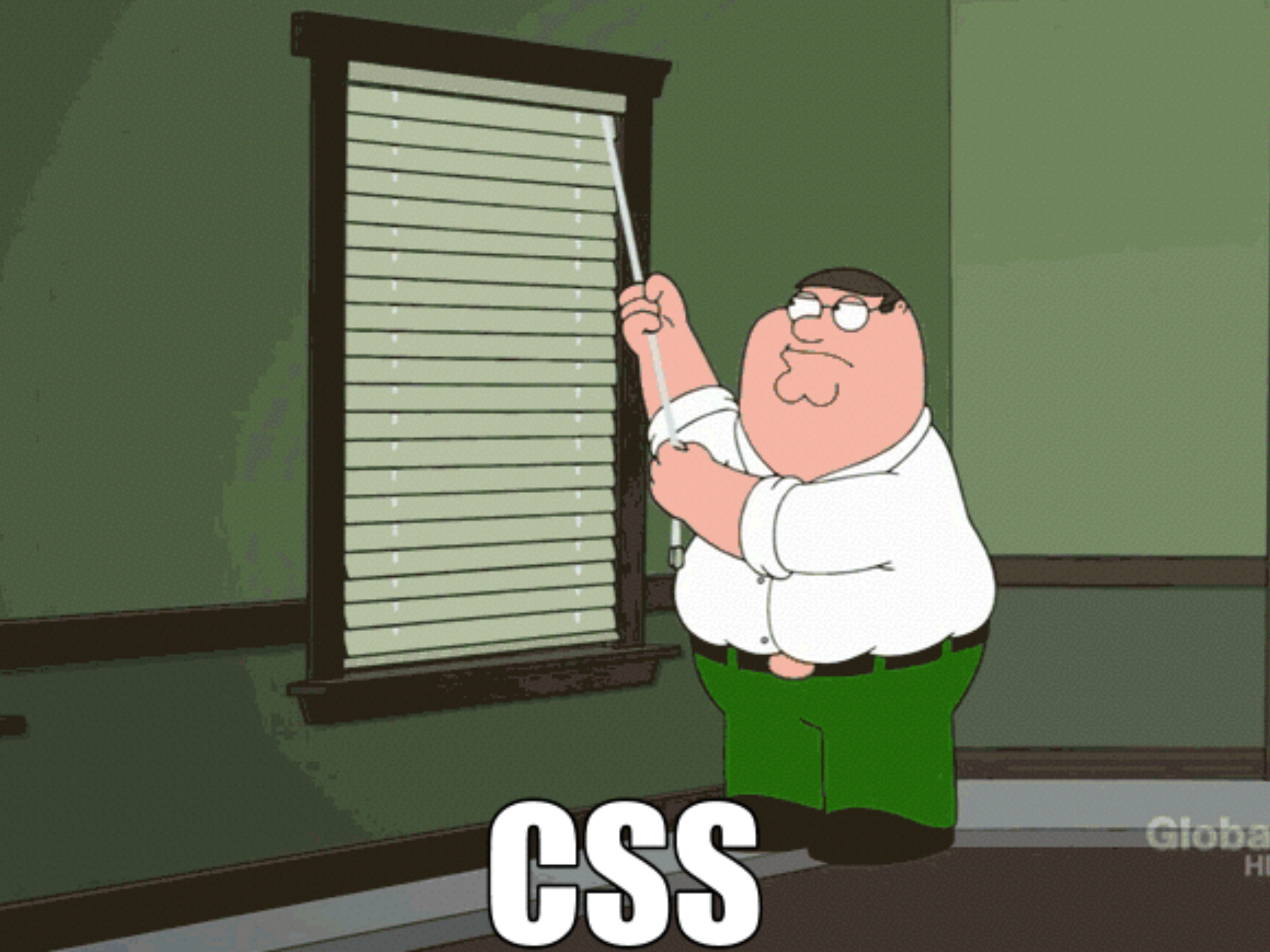
Our approach

- components
- fragments
- layouts
- pages



github.com/serverless/site/tree/master/src

http://bit.ly/phenomic-sites


CSS

What are these?

My Javascript fatique senses are tingling

speakerdeck.com/davidwells/bulletproof-css-in-react

Watch: bit.ly/postcss-talk

Localized Styles via PostCSS + CSS modules

```
/* Button.js React Component */
import React from 'react'
import styles from './Button.css'

class Button extends React.Component {
  render(){
    const text = this.props.text
    return (
      <div className={styles.wrapper}>
        <button className={styles.button}>
          {text}
        </button>
      </div>
    )
  }
}
```

CSS as per usual

```
/* Button.css */
.wrapper {
  font-size: 20px;
}

.button {
  border: 20px solid #bada55;
}
```

How CSS modules work

```
/* Button.js */
import React from 'react'
/* styles is imported as an object with classnames as keys */
import styles from './Button.css'
/* styles = {
    button: "Button__button___I_bKh"
    wrapper: "Button__wrapper___3fslE"
  }
*/
class Button extends React.Component {
  render(){
    const text = this.props.text
    return (
      <div className={styles.wrapper}>
        <button className={styles.button}>
          {text}
        </button>
      </div>
    )
  }
}
```

The end result

```
<!-- Localized Classes! -->
<div class="Button__wrapper___3fslE">
  <button class="Button__button___I_bKh">
    My button Text
  </button>
</div>
```

VS

```
<!-- collision city -->
<div class="wrapper">
  <button class="button">
    My button Text
  </button>
</div>
```

Global Assets

VARIABLES AND MIXINS

powered by JS!

Powered by PostCSS

```
var postCSSConfig = [
  /* enable mixins like Sass/Less */
  require('postcss-mixins')({
    mixins: require('./mixins')
  }),
  /* require global variables */
  require('postcss-simple-vars')({
    variables: function () {
      return require('./variables')
    },
    unknown: function (node, name, result) {
      node.warn(result, 'Unknown variable ' + name)
    }
  }),
]
```

```cpp
/* Global CSS variables for use in CSS and JS */
module.exports = {
  // -- Sizing
  unit: '1rem',
  navHeight: '65px',
  footerHeight: '155px',
  // -- Colors
  primary: '#ffd426',
  secondary: '#e55934',
  danger: '#aaa',
  // -- Indexes
  zIndexHighest: 300,
  zIndexHigher: 200,
  zIndexHigh: 100,
  zIndexNormal: 1,
  zIndexLow: -100,
  zIndexLower: -200,
  // -- Breakpoints
  breakpointXxs: '480px',
  breakpointXs: '600px',
  breakpointSmTablet: '720px',
  breakpointSm: '840px',
  breakpointMd: '960px',
  breakpointLgTablet: '1024px',
  breakpointLg: '1280px',
  breakpointXl: '1440px',
  breakpointXxl: '1600px',
  breakpointXxxl: '1920px',
}
```

```cpp
/* Global Mixins for use in CSS and JS */
module.exports = {
  /* Disable selectable text */
  noSelect: {
    '-webkit-touch-callout': 'none',
    '-webkit-user-select': 'none',
    '-khtml-user-select': 'none',
    '-moz-user-select': 'none',
    '-ms-user-select': 'none',
    'user-select': 'none'
  },
  OpenSans: {
    'font-family': 'Open Sans, sans-serif',
    'font-style': 'normal',
    'font-weight': 400,
    '-webkit-font-smoothing': 'antialiased',
    '-moz-osx-font-smoothing': 'grayscale',
  },
  // Function example
  image: function (mixin, path) {
    return {
      '&': {
        background: 'url(' + path + ')'
      },
      '@media (min-resolution: 120dpi)': {
        '&': {
          background: 'url(' + path + '@2x)'
        }
      }
    }
  }
}
```

Usage in CSS

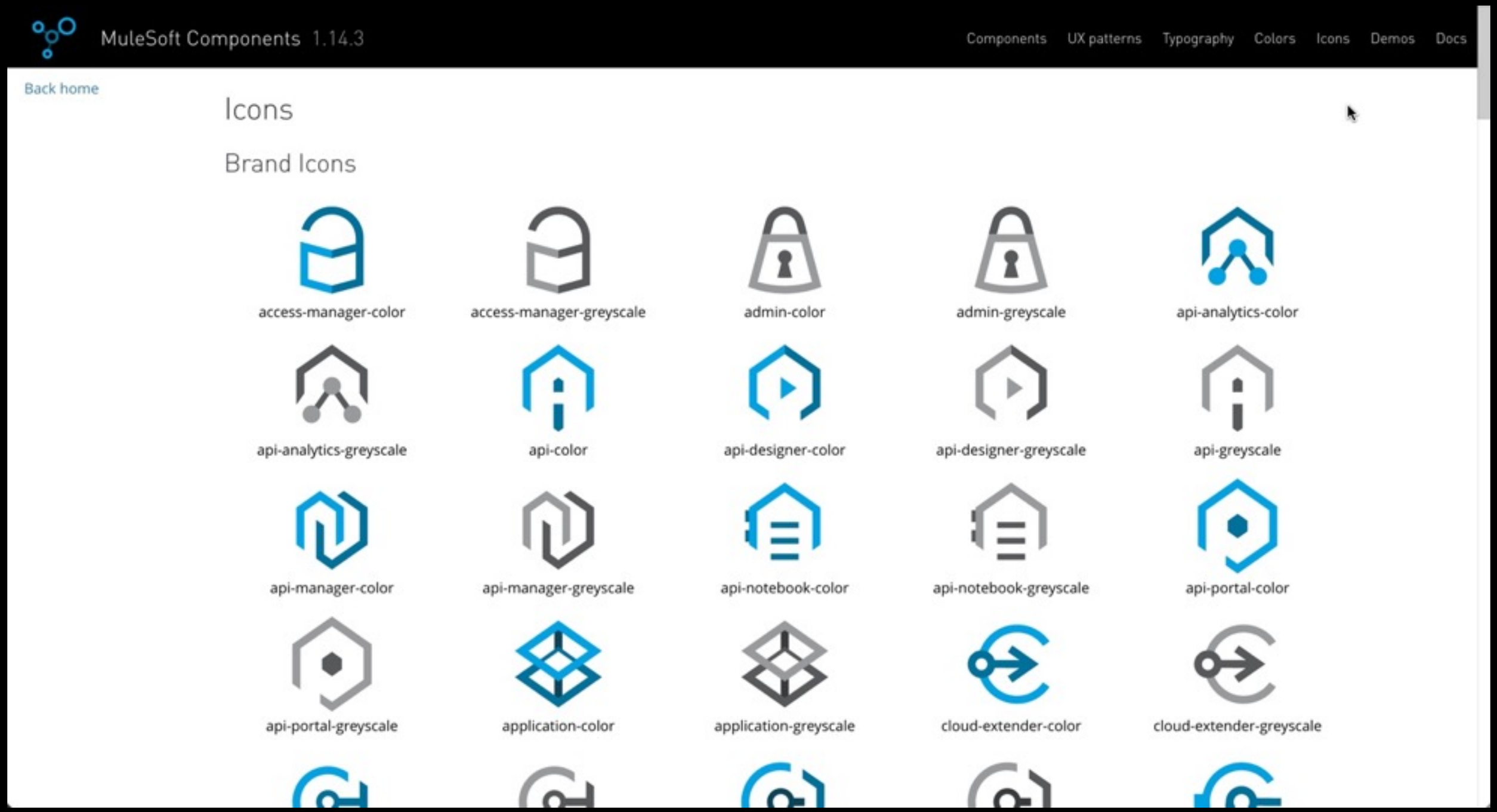
```
.hero {
  @mixin DinProLight; /* defined in mixins.js */
  color: $primaryBlue; /* defined in variables.js */
}
```

Usage in JS

```
import React from 'react'
import variables from 'path/to/global-variables'

class Button extends React.Component {
  render(){
    const styles = {
      color: variables.primary
      zIndex: variables.aboveModal
    }
    return (
      <div style={styles}>
        WOW!
      </div>
    )
  }
}
```

Icons


MuleSoft Components 1.14.3
Components
UX patterns
Typography
Colors
Icons
Demos
Docs
Back home
Icons
Brand Icons
access-manager-color
access-manager-greyscale
admin-color
admin-greyscale
api-analytics-color
api-analytics-greyscale
api-color
api-designer-color
api-designer-greyscale
api-greyscale
api-manager-color
api-manager-greyscale
api-notebook-color
api-notebook-greyscale
api-portal-color
api-portal-greyscale
application-color
application-greyscale
cloud-extender-color
cloud-extender-greyscale

1. Designers make SVGs
2. build script runs all svgs through webpack
3. the 'webpack-svgstore-plugin' plugin optimizes svgs and creates a sprite
4. Sprite gets injected into the DOM
5. Profit $$$

```
<svg><use id='icon-xyz'></svg>
```

Distribution

webpack

MODULE BUNDLER

+

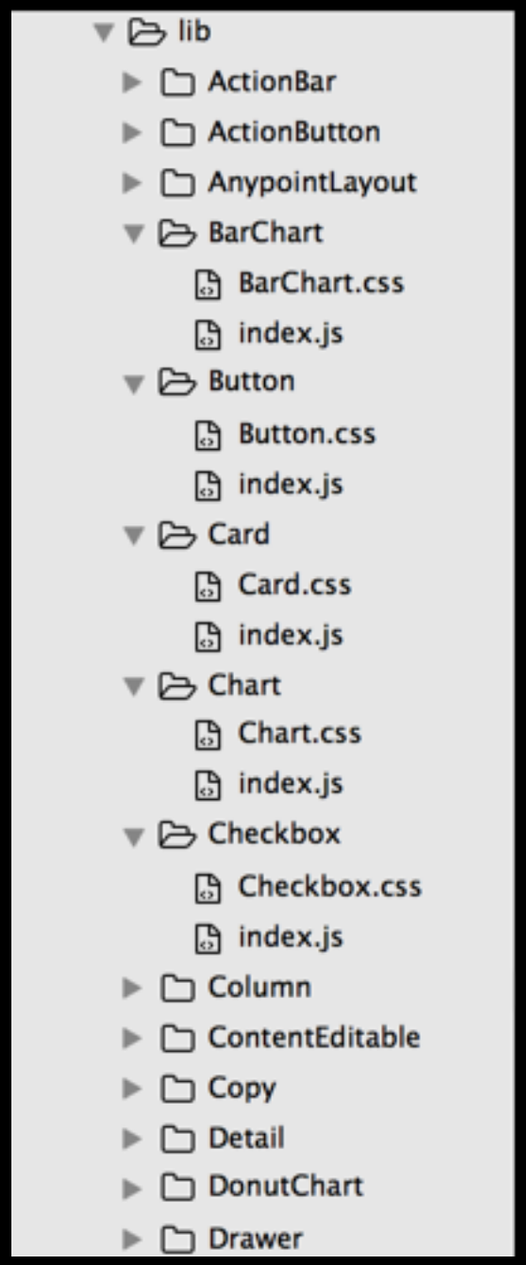
npm

Build components separately

Asset	Size	Chunks		Chunk Names
Checkbox/index.js	4.13 kB	65	[emitted]	Checkbox
index/index.js	441 kB	0	[emitted]	index
TypeAhead/index.js	119 kB	2	[emitted]	TypeAhead
Select/index.js	21.8 kB	3	[emitted]	Select
Modal/index.js	11.5 kB	4	[emitted]	Modal
ToolbarHeader/index.js	4.06 kB	5	[emitted]	ToolbarHeader
Filter/index.js	6.14 kB	6	[emitted]	Filter
Detail/index.js	10.2 kB	7	[emitted]	Detail
TextField/index.js	10.8 kB	8	[emitted]	TextField
Table/index.js	11 kB	9	[emitted]	Table
SideBar/index.js	3.96 kB	10	[emitted]	SideBar
AnypointLayout/index.js	4.62 kB	11	[emitted]	AnypointLayout
ViewSwitcher/index.js	3.41 kB	12	[emitted]	ViewSwitcher
TimePicker/index.js	7.3 kB	13	[emitted]	TimePicker
TextArea/index.js	7.63 kB	14	[emitted]	TextArea
MediaObject/index.js	4.24 kB	15	[emitted]	MediaObject
FilterBar/index.js	2.73 kB	16	[emitted]	FilterBar
DropdownToggle/index.js	3.81 kB	17	[emitted]	DropdownToggle
DropdownMenu/index.js	5.41 kB	18	[emitted]	DropdownMenu
Chart/index.js	5.87 kB	19	[emitted]	Chart
Card/index.js	6.05 kB	20	[emitted]	Card
Drawer/index.js	6.13 kB	21	[emitted]	Drawer
Button/index.js	5.41 kB	22	[emitted]	Button
Icon/index.js	103 kB	23	[emitted]	Icon
Toggle/index.js	4.57 kB	24	[emitted]	Toggle
Search/index.js	3.57 kB	25	[emitted]	Search

```
import Button from '@component-lib/lib/Button'
```

built Button/index.js




```
/***/ 4:
/***/ function(module, exports) {

  module.exports = require("../Icon");

/***/ },

/***/ 34:
/***/ function(module, exports) {

  module.exports = require("../Tooltip");

/***/ },
```

```
/***/ 320:
/***/ function(module, exports) {

  module.exports = require("../Button/Button.css");

/***/ }
```

Monitor Component Usage

```
webpack built 60f444f82f6388ccfc44 in 5227ms
ping server with results
project landing-page-ui
project version 0.0.1
components used [ 'Button',
  'Drawer',
  'Flex',
  'Icon',
  'Image',
  'List',
  'ListItem',
  'MediaObject',
  'NavBar',
  'Popover',
  'Row',
  'Tooltip' ]
component count 12
```

```
var lookup = require('climb-lookup')
var packageJSONPath = lookup.lookupSync('package.json')
var packageInfo = require(packageJSONPath)

function ComponentUsagePlugin (output, options) {
  this.options = options
}

ComponentUsagePlugin.prototype.apply = function apply (compiler) {
  compiler.plugin('done', function (stats) {
    var data = stats.toJson()
    // data has all files being used in project
    console.log(data)

    // Parse for component files
    // var componentsUsed = getComponentsUsed(data)

    // Send to serverless function
    // sendToAWSLambda(packageInfo, componentsUsed)
  })
}

module.exports = ComponentUsagePlugin
```

send usage stats to lambda function via SERVERLESS

lessons learned

- be vigilant with your component APIs. Prop sprawl happens and can be avoided with higher order components
- don't add things before there are specific use cases for them.
- name things well and agree on standard before hand
- when in doubt on future use cases provide custom component renderers
- Setup linting first!!!!!!
- Test server side renderablity early and often
- add linting for a11y accessibility

Component Rules (aka PropTypes conventions)

- Keep things minimal
- Mirror DOM where possible
- Follow Good Naming Conventions
- Add docBlock Comments above propTypes
- Avoid Prop Sprawl
- No unnecessary abstractions

davidwells.io/react-prop-type-best-practices

Final thoughts

- with this setup you can achieve a faster development velocity
- individuals can work on isolated parts of the system without fear of breaking other things
- you have a multi platform UI library
- you can render static apps (or server side if you wish) for better UX + indexability
- choosing the static path gives you out of the box raw scale power with less devop complexity

Future
Stack

BABEL

npm

webpack
MODULE BUNDLER

PostCSS

ES6

Phenomic

CSS
MODULES

Redux

SERVERLESS