OVHcloud

FrontSide

Design, develop and manage a catalog of Web Components

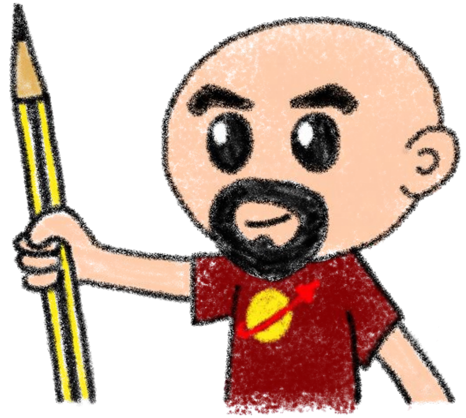
Horacio Gonzalez

2020-10-15

@LostInBrittany

Who are we?

Introducing myself and introducing ~~OVH~~ OVHcloud



Horacio Gonzalez

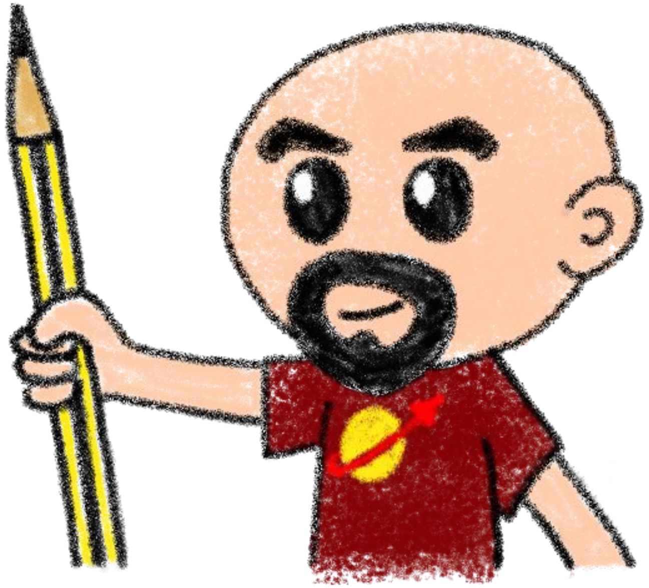
FrontSide

@LostInBrittany

Spaniard lost in Brittany, developer, dreamer and all-around geek

OVHcloud

DevRel Leader



Le Halloween des Speakers

Finist Devs

DevFest du Bout du Monde

Google Developers Experts 2019

Web Technologies GDE

Flutter

OVHcloud: A global leader

Web Cloud & Telcom

Private Cloud

Public Cloud

Storage

Network & Security

30 Data Centers
in 12 locations

34 Points of Presence
on a 20 TBPS Bandwidth Network

2200 Employees
worldwide

115K Private Cloud
VMS running

300K Public Cloud
instances running

380K Physical Servers
running in our data centers

1 Million+ Servers
produced since 1999

1.5 Million Customers
across 132 countries

3.8 Million Websites
hosting

1.5 Billion Euros Invested
since 2016

P.U.E. 1.09
Energy efficiency indicator

20 Years in Business
Disrupting since 1999

The 3 minutes context

What the heck are web component?

Web Components

Web standard W3C

Web Components

Available in all modern browsers:

Firefox, Safari, Chrome

Web Components

Create your own HTML tags

Encapsulating look and behavior

Web Components

Fully interoperable

With other web components, with any framework

Web Components

CUSTOM ELEMENTS

SHADOW DOM

TEMPLATES

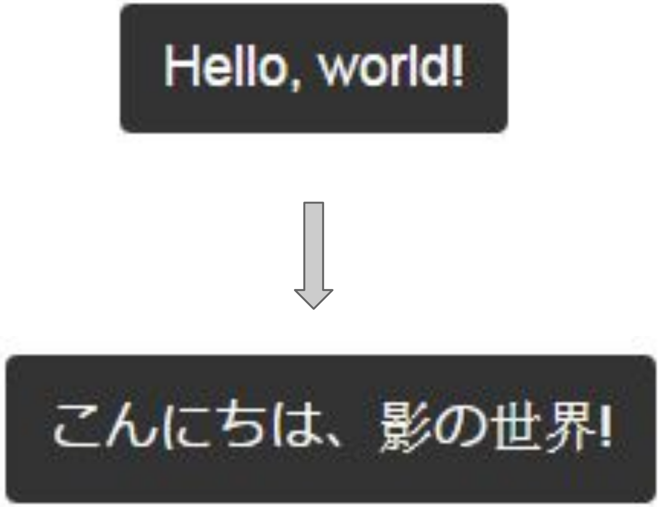
Custom Element

To define your own HTML tag

```
<body>
  ...
  <script>
    window.customElements.define('my-element',
      class extends HTMLElement {...});
  </script>
  <my-element></my-element>
</body>
```

Shadow DOM

To encapsulate subtree and style in an element



```
<button>Hello, world!</button>
<script>
var host = document.querySelector('button');
const shadowRoot = host.attachShadow({mode:'open'});
shadowRoot.textContent = 'こんにちは、影の世界!';
</script>
```

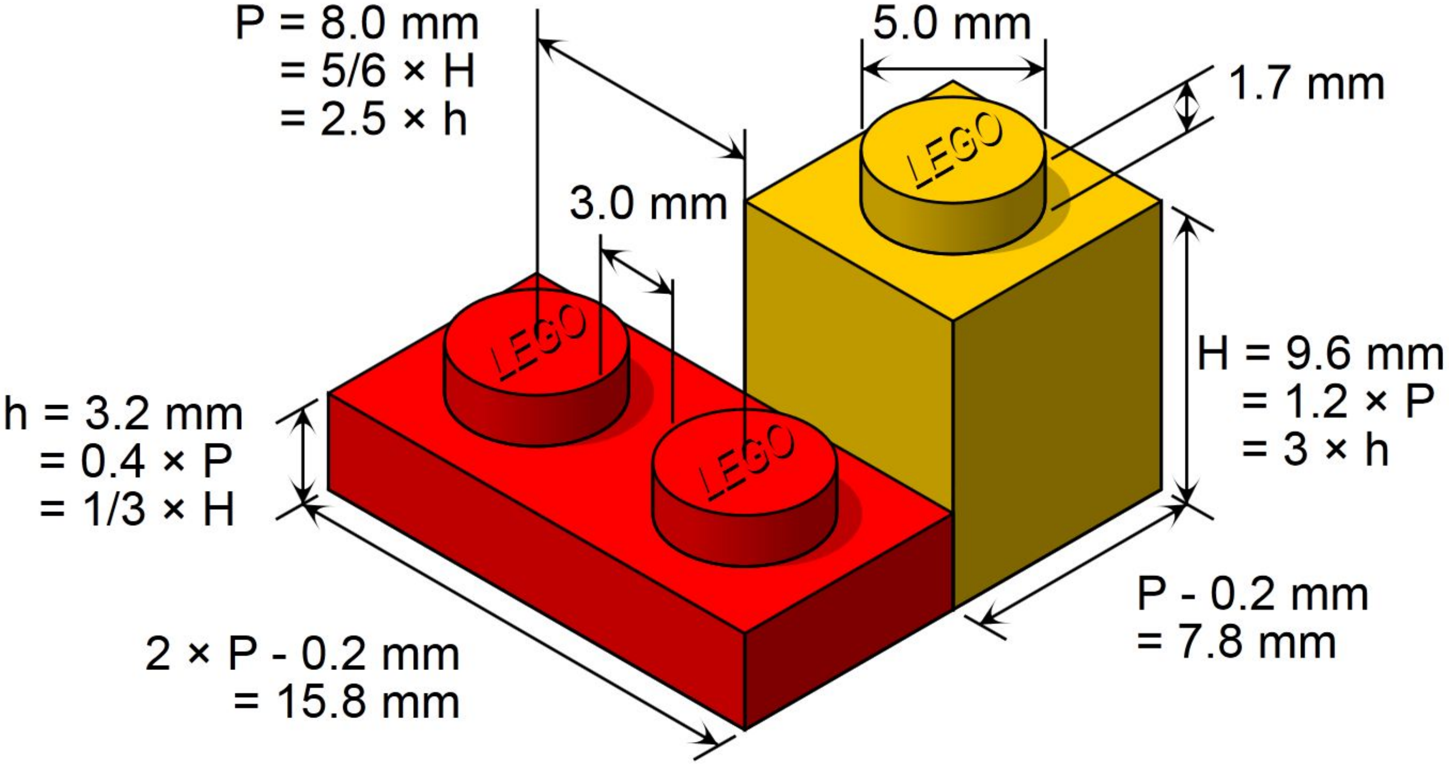
Template

To have clonable document template

```
<template id="mytemplate">
  <img src="" alt="great image">
  <div class="comment"></div>
</template>

var t = document.querySelector('#mytemplate');
// Populate the src at runtime.
t.content.querySelector('img').src = 'logo.png';
var clone = document.importNode(t.content, true);
document.body.appendChild(clone);
```

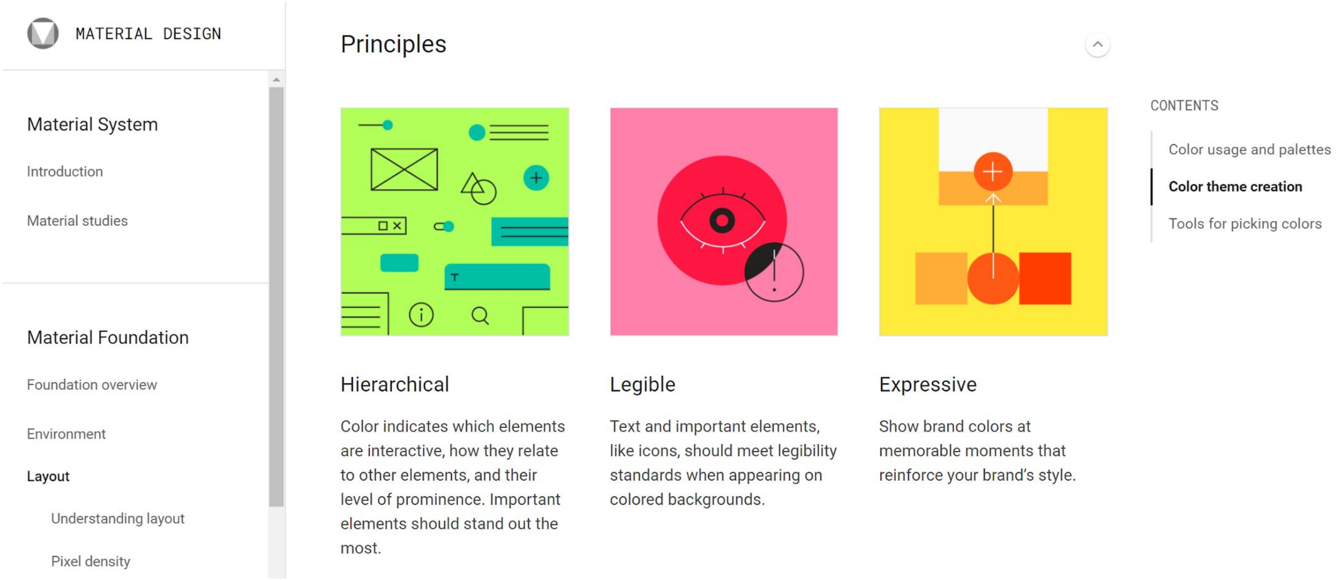
But in fact, it's just an element…

- Attributes
- Properties
- Methods
- Events



So, what are Design Systems?

And why should I look at them?



MATERIAL DESIGN

Material System

Introduction

Material studies

Material Foundation

Foundation overview

Environment

Layout

Understanding layout

Pixel density

Principles

Hierarchical

Color indicates which elements are interactive, how they relate to other elements, and their level of prominence. Important elements should stand out the most.

Legible

Text and important elements, like icons, should meet legibility standards when appearing on colored backgrounds.

Expressive

Show brand colors at memorable moments that reinforce your brand's style.

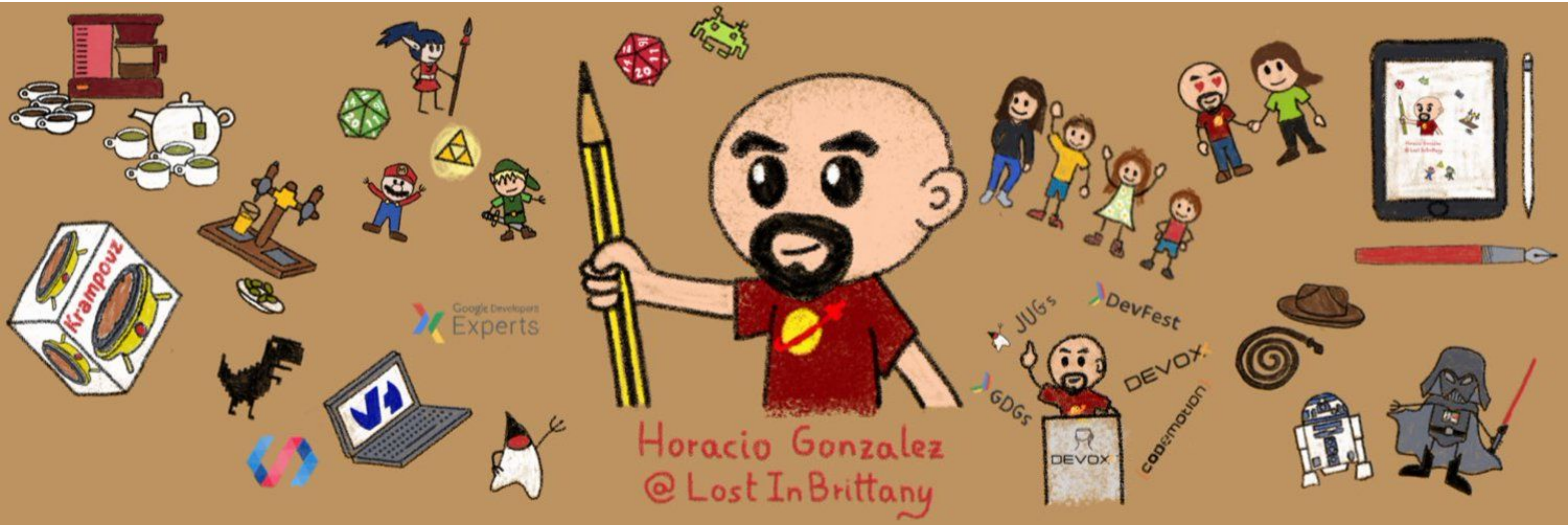
CONTENTS

Color usage and palettes

Color theme creation

Tools for picking colors

A talk for devs by a dev



I am not a designer, neither I play one on TV...

The same or different?

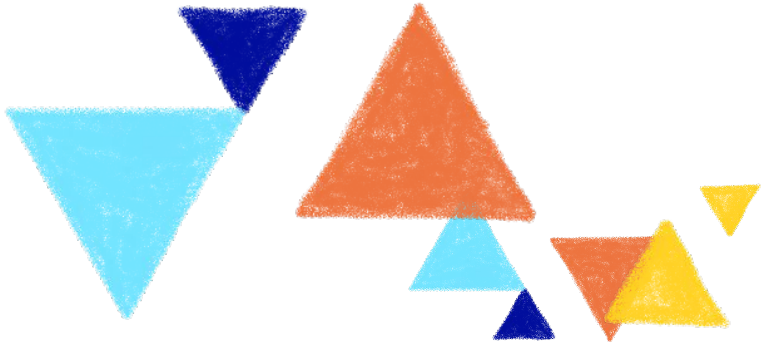
Design System

Component Catalog

Style Guide

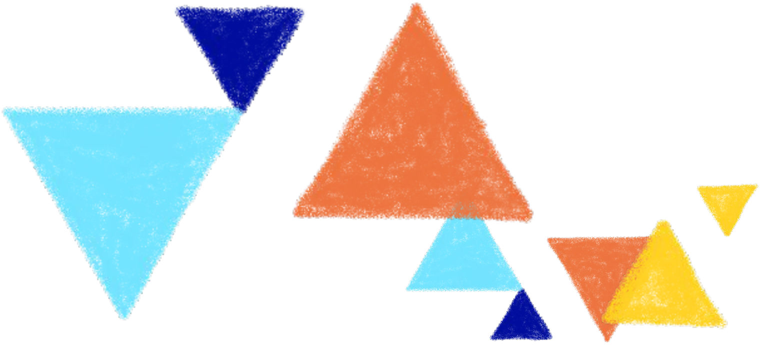
Style Guides

A **document** listing the **styles**, **patterns**, **practices**, and **principles** of a brand **design standards**



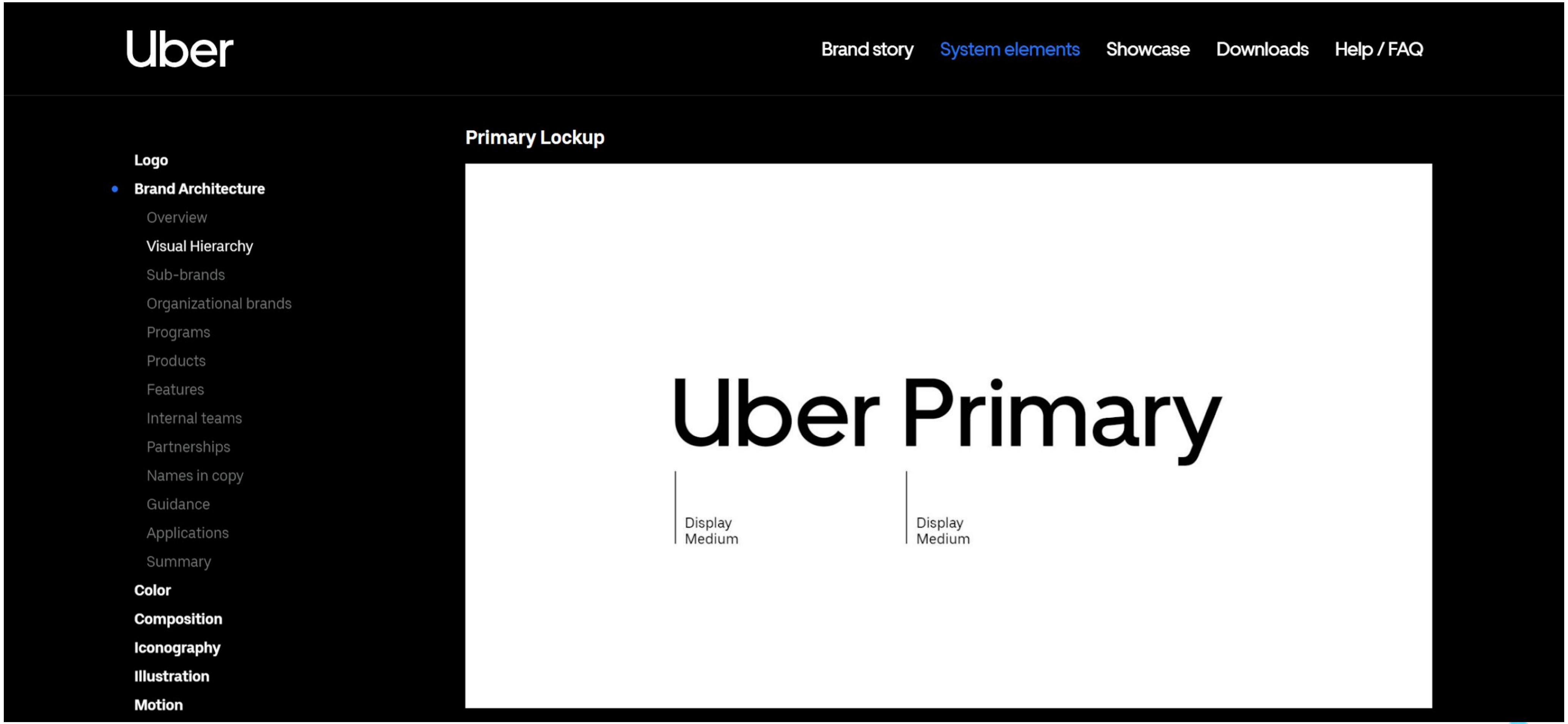
Style Guides

Style guides define the **application’s look and feel**



Style Guide Example: Uber

FrontSide



Uber

Brand story · System elements · Showcase · Downloads · Help / FAQ

- Logo
- Brand Architecture
  - Overview
  - Visual Hierarchy
  - Sub-brands
  - Organizational brands
  - Programs
  - Products
  - Features
  - Internal teams
  - Partnerships
  - Names in copy
  - Guidance
  - Applications
  - Summary
- Color
- Composition
- Iconography
- Illustration
- Motion

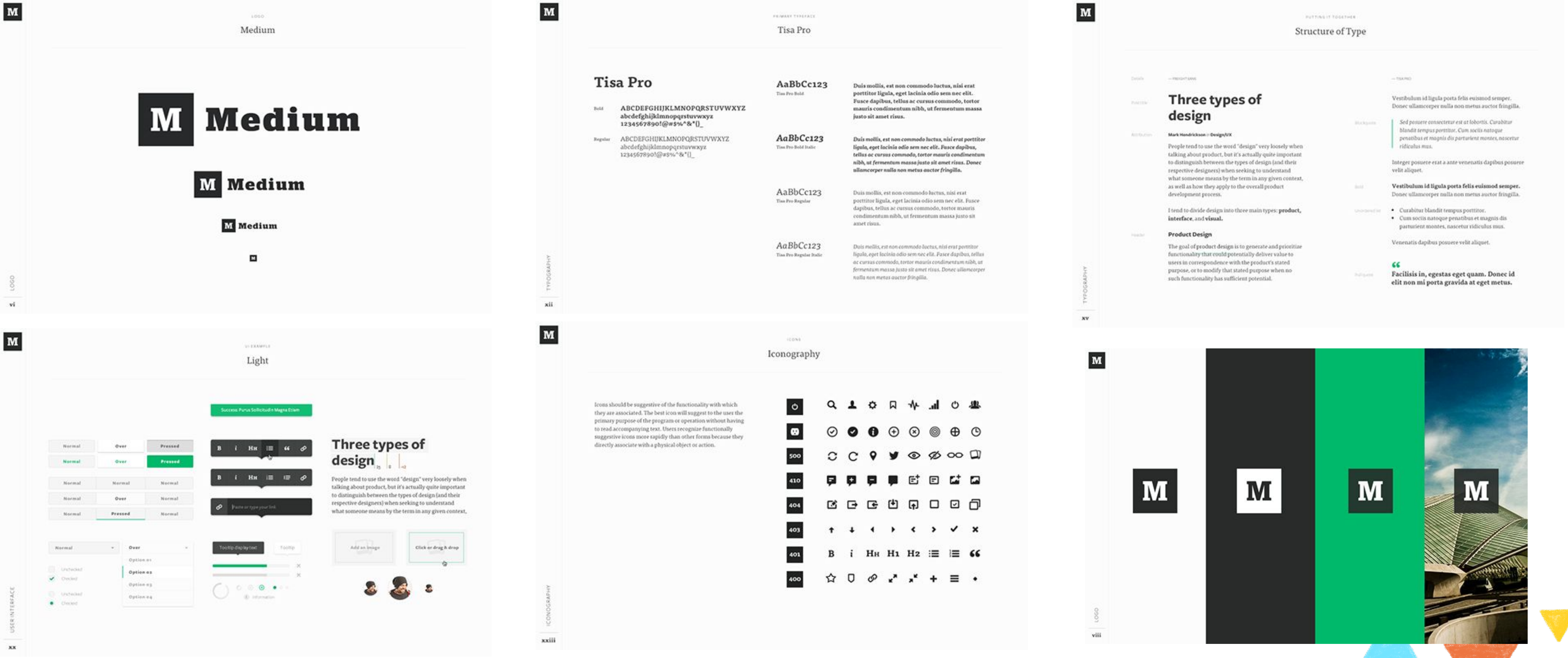
Primary Lockup

Uber Primary

Display Medium · Display Medium

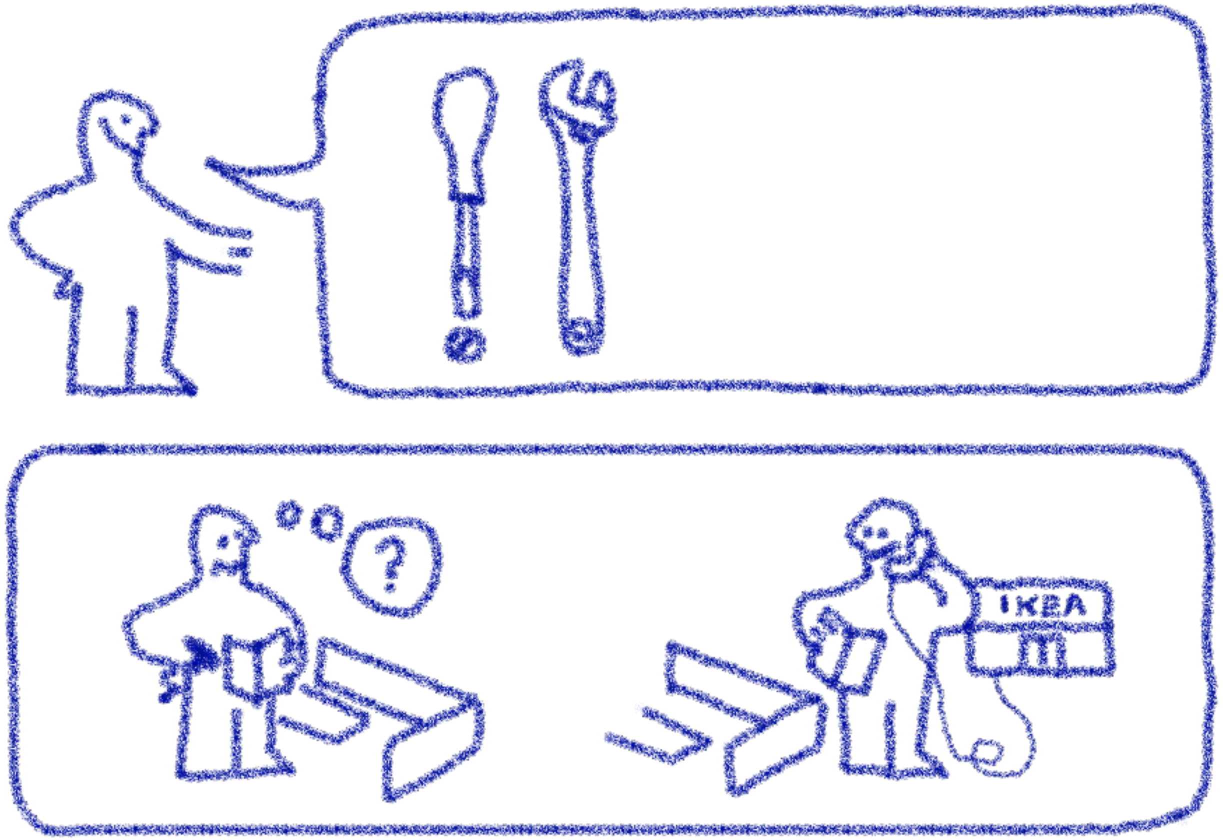
https://brand.uber.com/

Style Guide Example: Medium



https://www.behance.net/gallery/7226653/Medium-Brand-Development

Style Guides alone are ambiguous

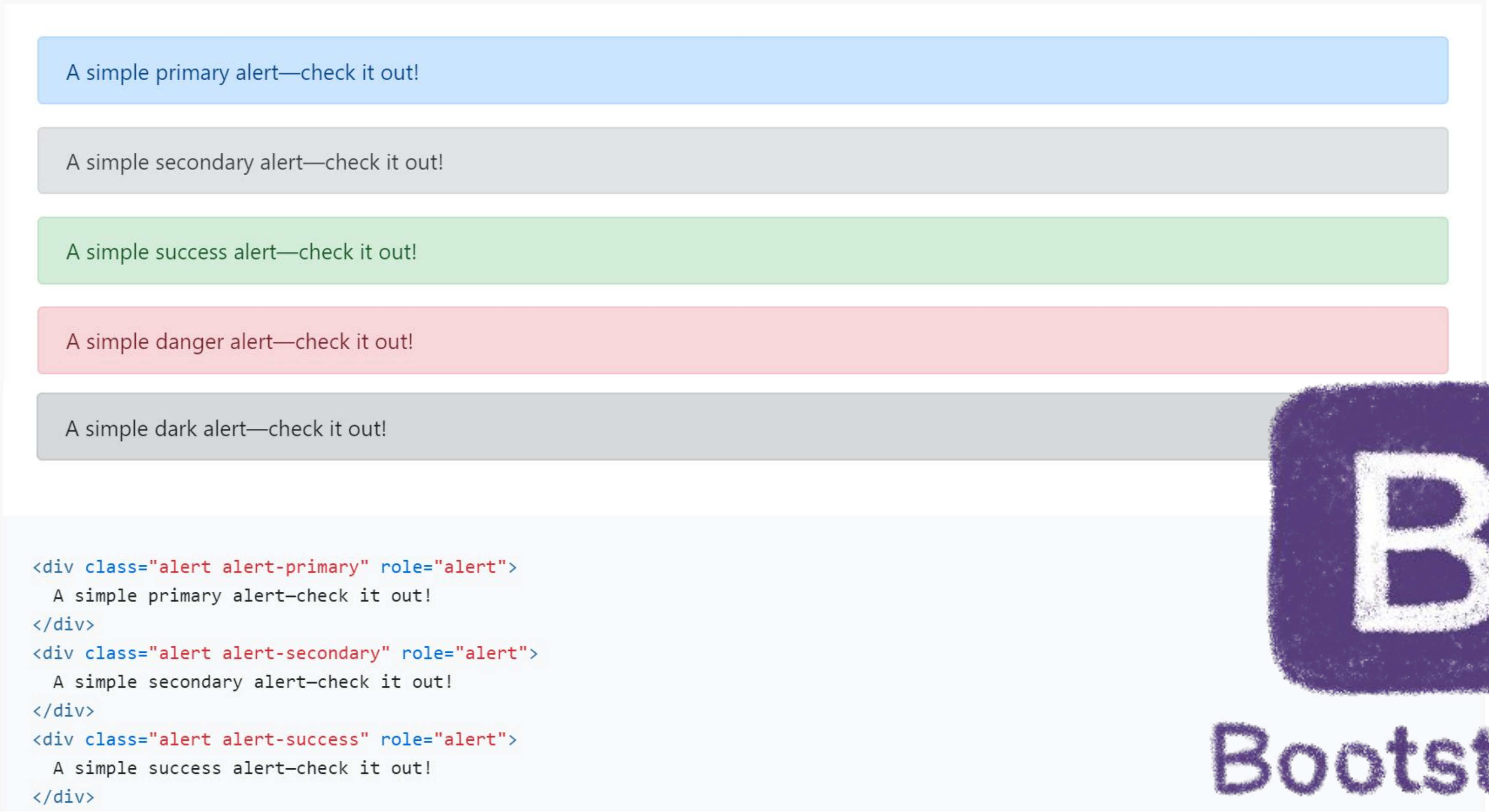


Interpretation needed to adapt
the preconisation to the use case

Component Catalogs

A **component catalog** is a **repository** of components, with one or several **implementations**, code **examples** and **technical documentation**

Component Catalog example: Bootstrap



A simple primary alert—check it out!

A simple secondary alert—check it out!

A simple success alert—check it out!

A simple danger alert—check it out!

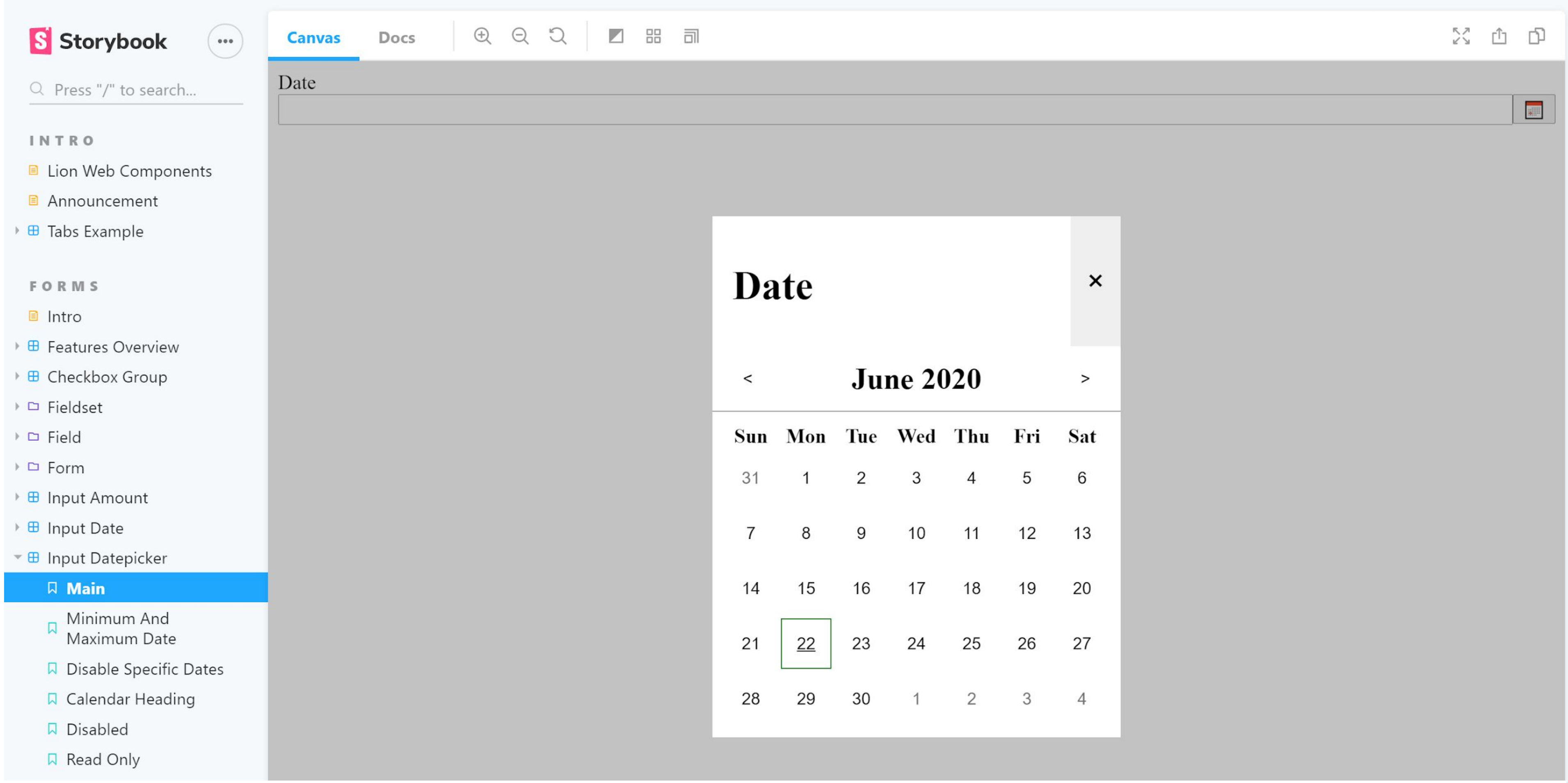
A simple dark alert—check it out!

```
<div class="alert alert-primary" role="alert">
  A simple primary alert—check it out!
</div>
<div class="alert alert-secondary" role="alert">
  A simple secondary alert—check it out!
</div>
<div class="alert alert-success" role="alert">
  A simple success alert—check it out!
</div>
```

Bootstrap

https://getbootstrap.com/

Component Catalog Example: ING's Lion



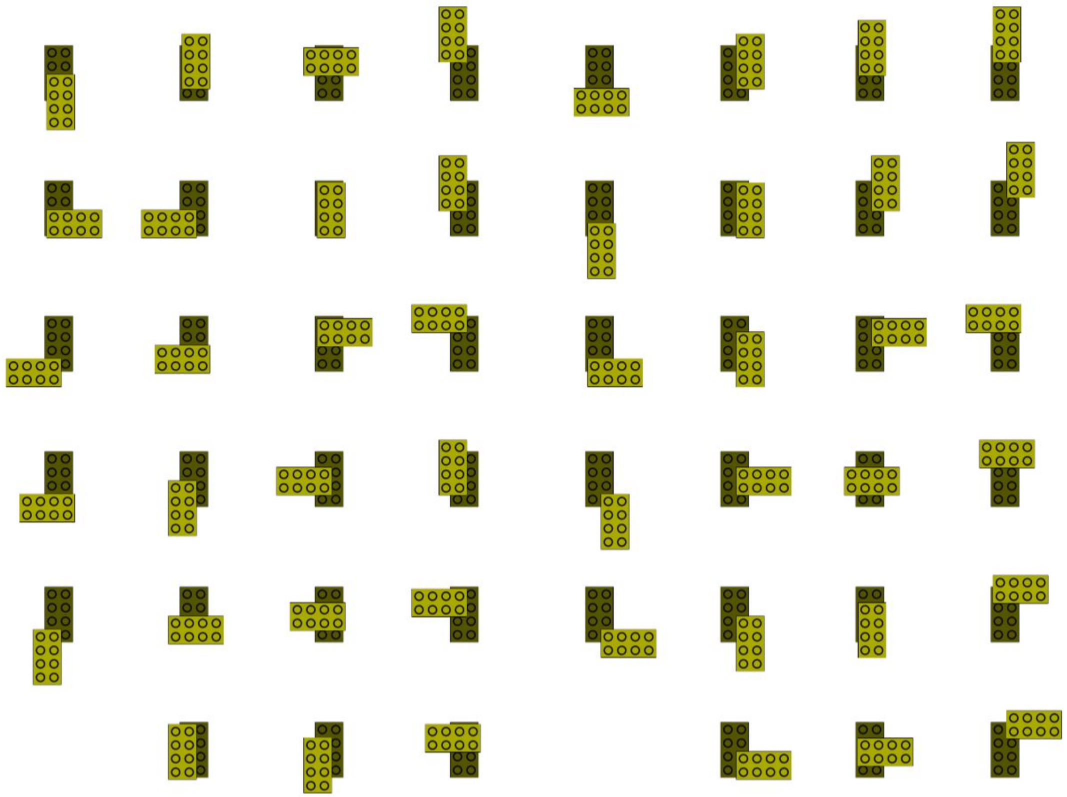
Date

June 2020

Sun	Mon	Tue	Wed	Thu	Fri	Sat
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	1	2	3	4

https://lion-web-components.netlify.app/

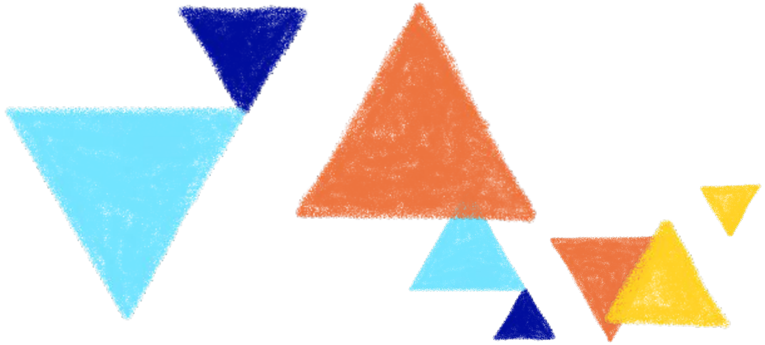
Catalogs alone create inconsistency



Like using the same LEGO bricks to create very different objects

Design Systems

A Design System is like a **common visual language** for **product teams**



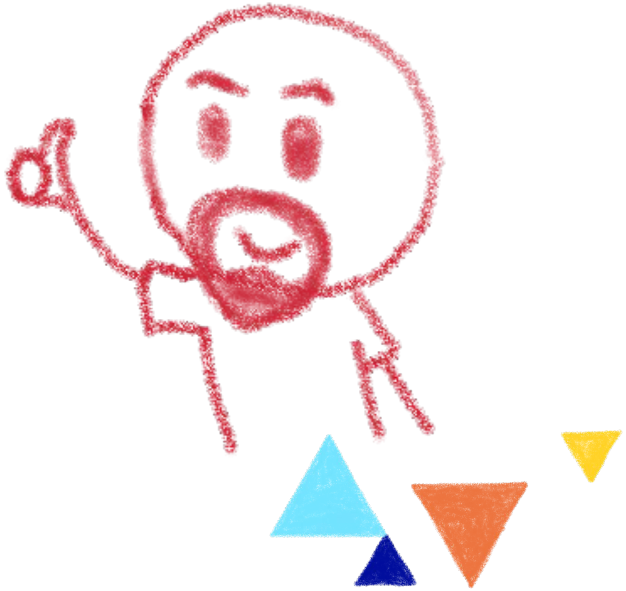
Design systems

A Design System is a set of **design standards**, **documentations**, and **principles**, alongside with the toolkit (**UI patterns** and **code components**) to achieve those standards

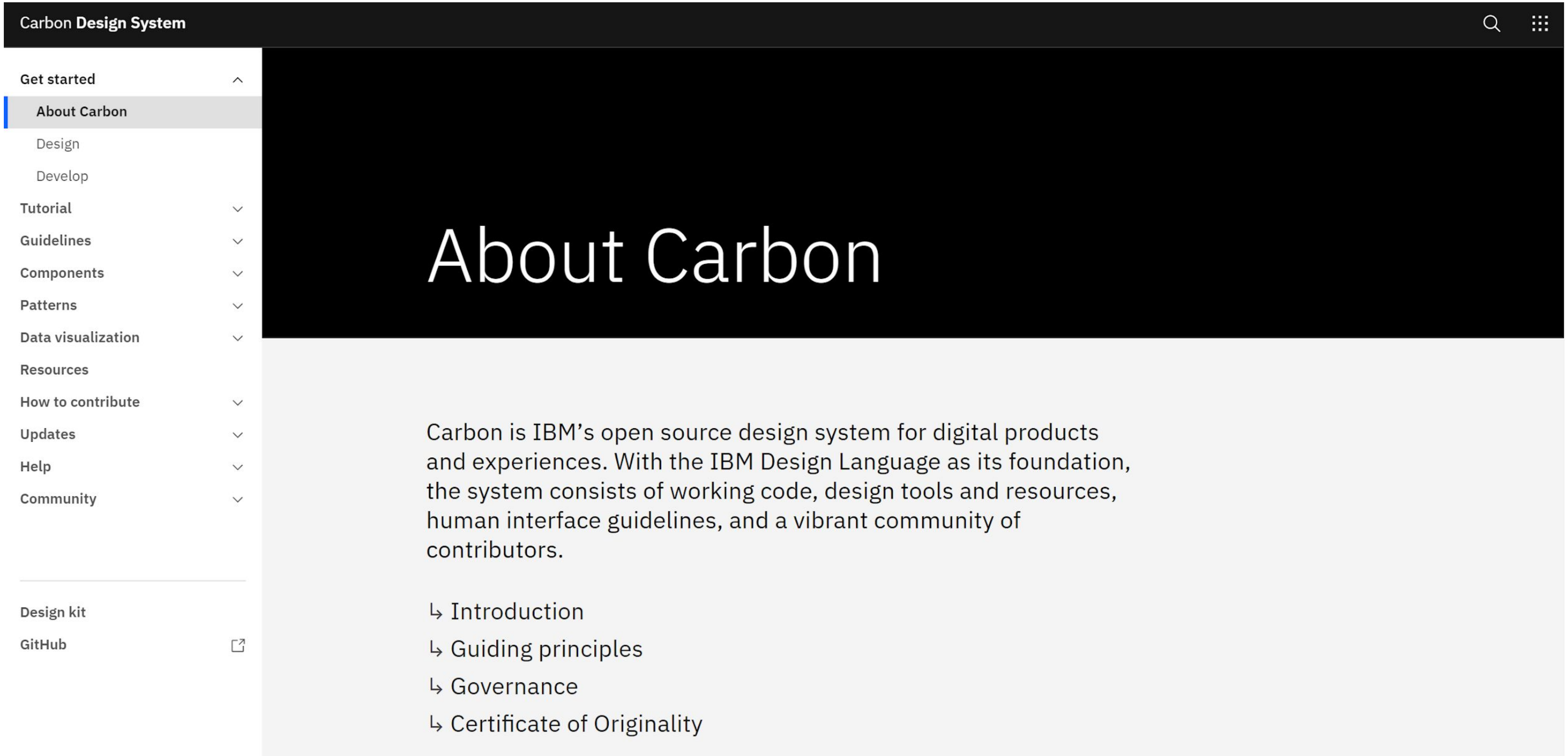
Design systems

Design System ≈

Style Guide + Component Catalog



Example: Carbon Design System



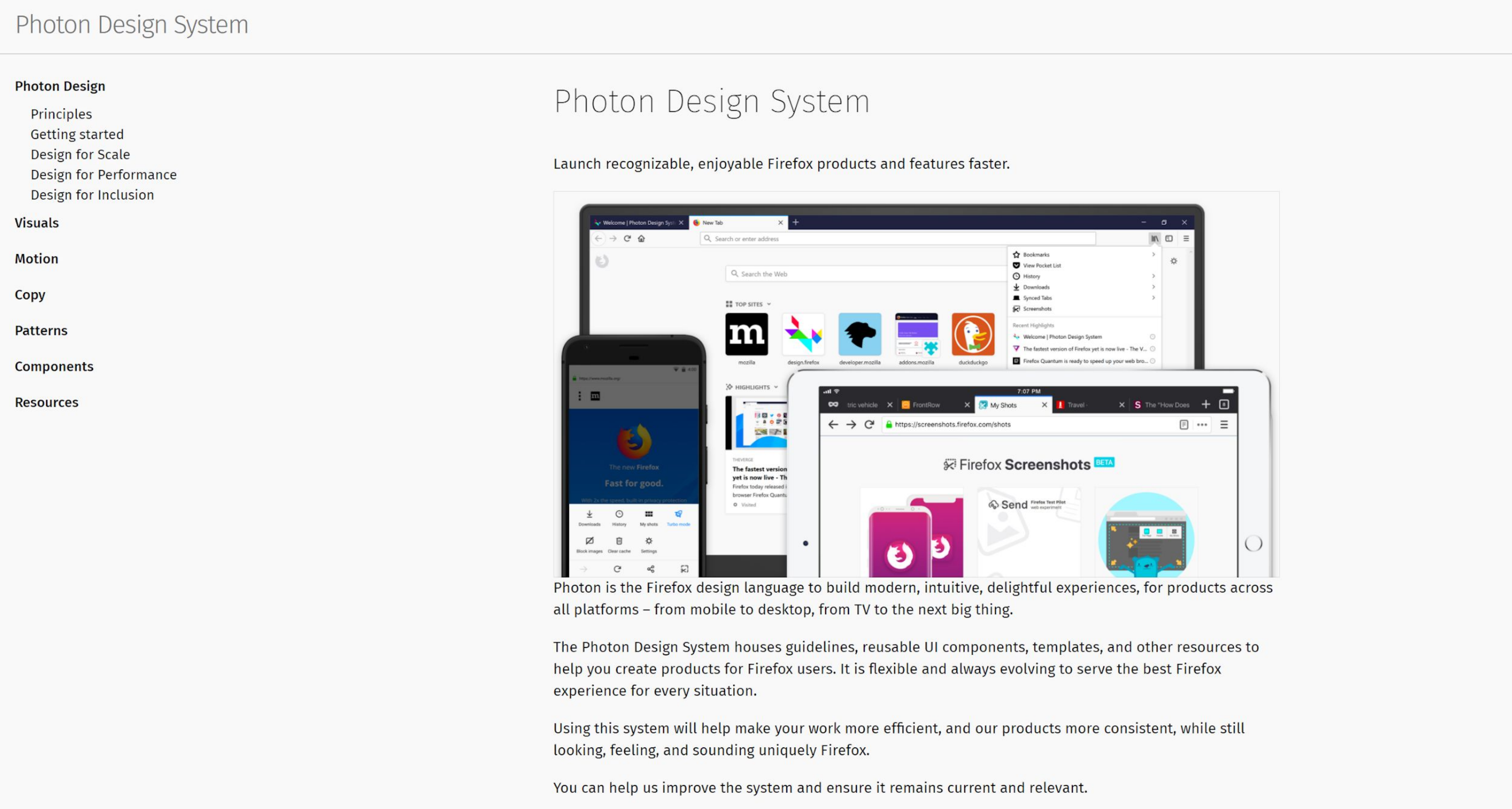
About Carbon

Carbon is IBM’s open source design system for digital products and experiences. With the IBM Design Language as its foundation, the system consists of working code, design tools and resources, human interface guidelines, and a vibrant community of contributors.

↳ Introduction
↳ Guiding principles
↳ Governance
↳ Certificate of Originality

https://www.carbondesignsystem.com/

Example: Firefox's Photon Design System



Photon Design System

Photon Design

- Principles
- Getting started
- Design for Scale
- Design for Performance
- Design for Inclusion

Visuals

Motion

Copy

Patterns

Components

Resources

Photon Design System

Launch recognizable, enjoyable Firefox products and features faster.

Photon is the Firefox design language to build modern, intuitive, delightful experiences, for products across all platforms – from mobile to desktop, from TV to the next big thing.

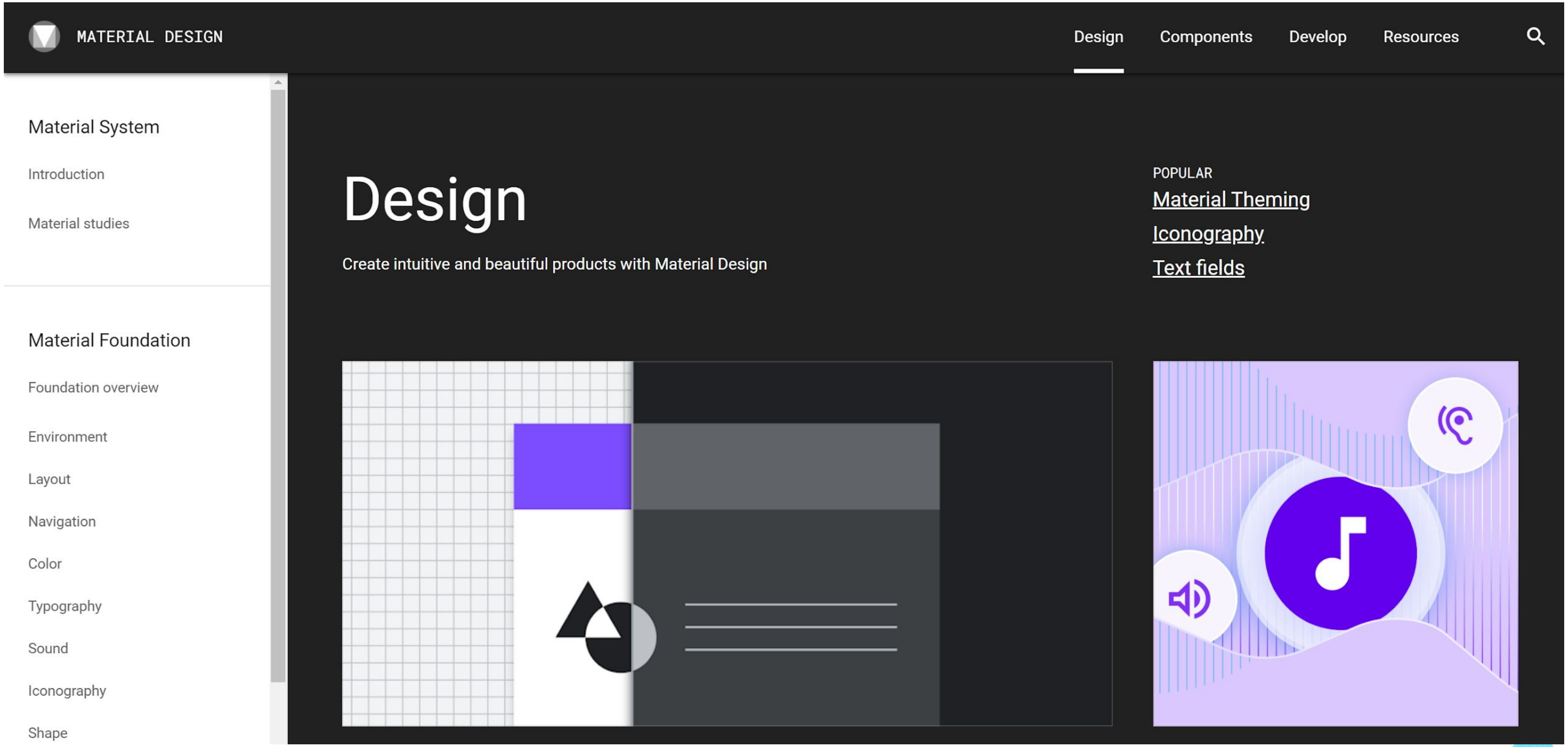
The Photon Design System houses guidelines, reusable UI components, templates, and other resources to help you create products for Firefox users. It is flexible and always evolving to serve the best Firefox experience for every situation.

Using this system will help make your work more efficient, and our products more consistent, while still looking, feeling, and sounding uniquely Firefox.

You can help us improve the system and ensure it remains current and relevant.

https://design.firefox.com/photon/

Example: Material Design



MATERIAL DESIGN

Design Components Develop Resources

Material System

Introduction

Material studies

Material Foundation

Foundation overview

Environment

Layout

Navigation

Color

Typography

Sound

Iconography

Shape

Design

Create intuitive and beautiful products with Material Design

POPULAR

Material Theming

Iconography

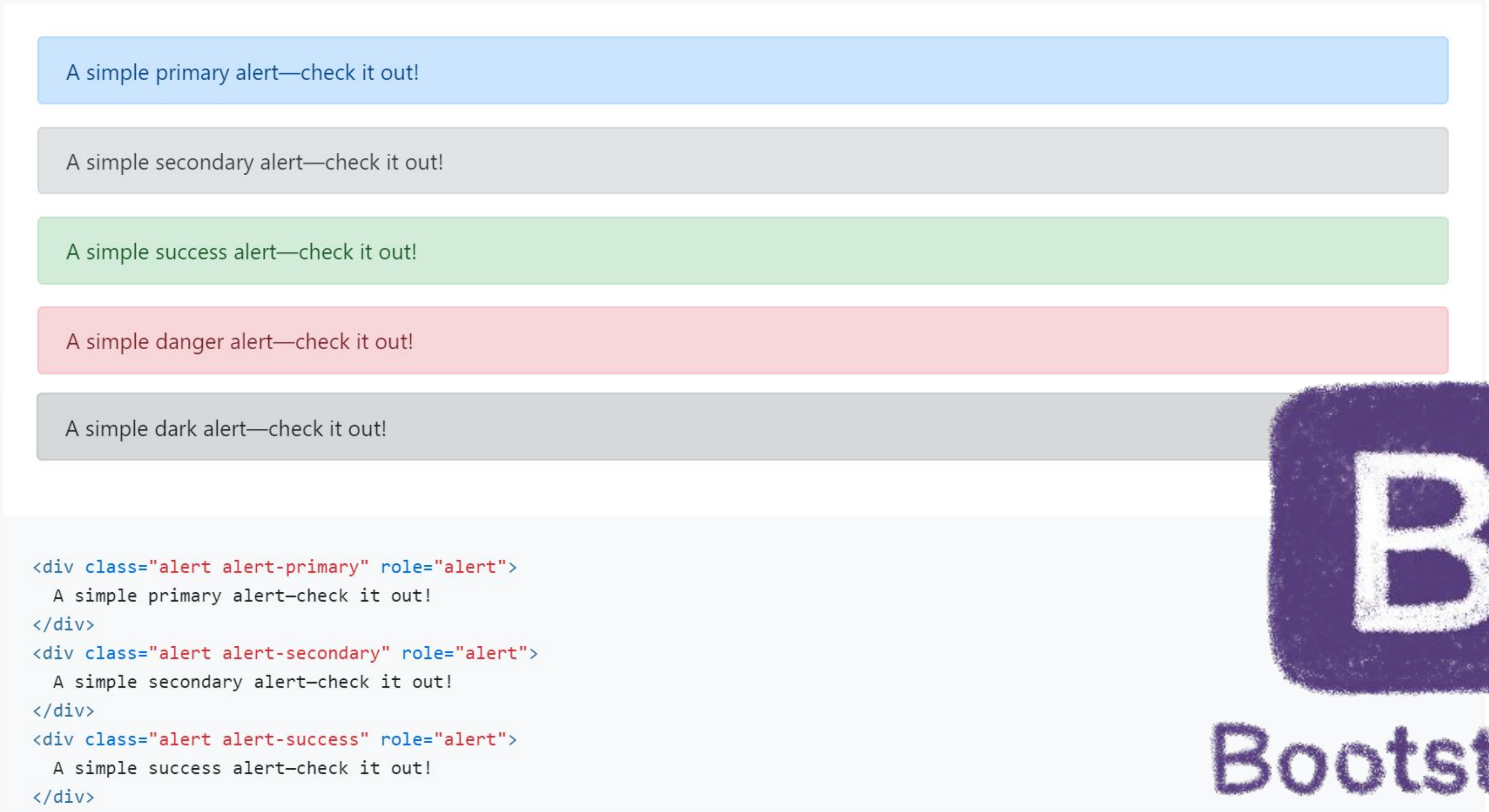
Text fields

https://material.io/

The component catalog

The poor relative of the Design System family

Let's choose a simple example



A simple primary alert—check it out!

A simple secondary alert—check it out!

A simple success alert—check it out!

A simple danger alert—check it out!

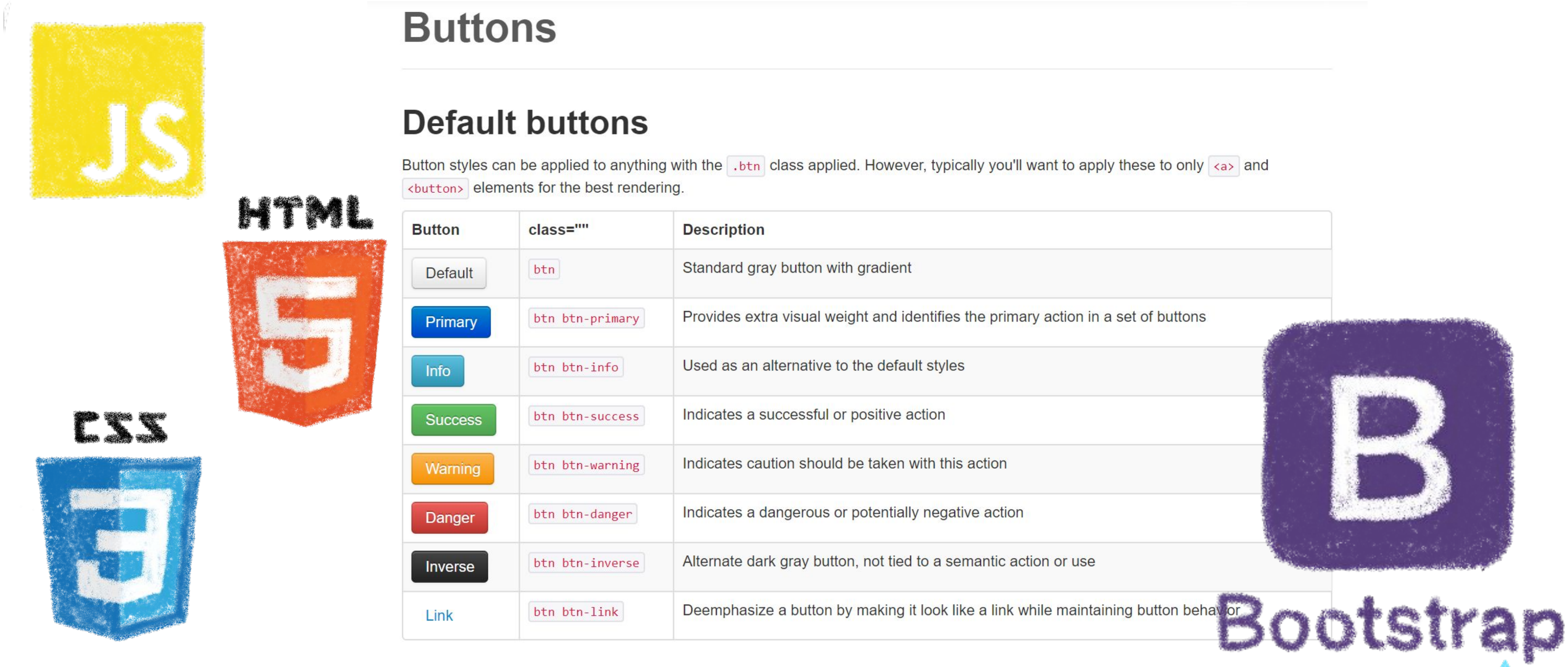
A simple dark alert—check it out!

```
<div class="alert alert-primary" role="alert">
  A simple primary alert—check it out!
</div>
<div class="alert alert-secondary" role="alert">
  A simple secondary alert—check it out!
</div>
<div class="alert alert-success" role="alert">
  A simple success alert—check it out!
</div>
```

Bootstrap

Bootstrap based component catalogs

A long time ago



Buttons

Default buttons

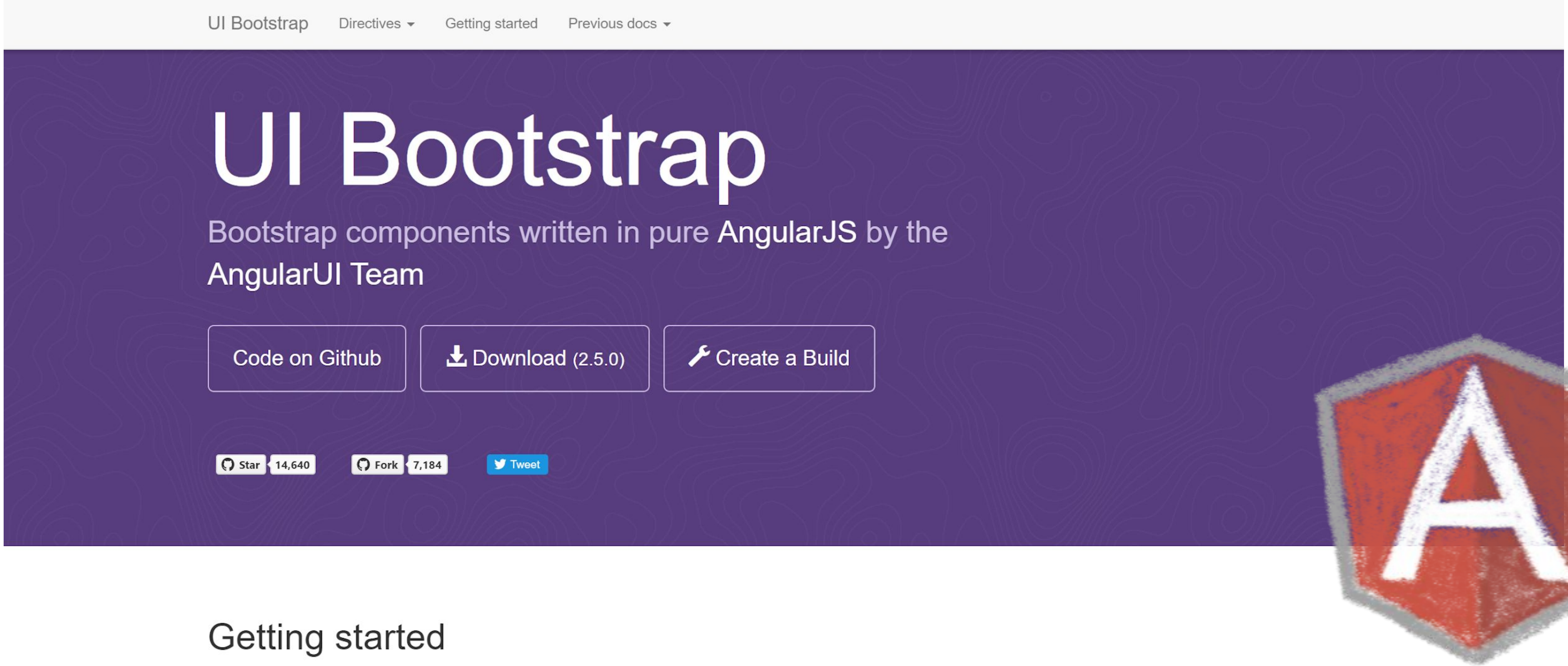
Button styles can be applied to anything with the `.btn` class applied. However, typically you'll want to apply these to only `<a>` and `<button>` elements for the best rendering.

Button	class=""	Description
Default	`btn`	Standard gray button with gradient
Primary	`btn btn-primary`	Provides extra visual weight and identifies the primary action in a set of buttons
Info	`btn btn-info`	Used as an alternative to the default styles
Success	`btn btn-success`	Indicates a successful or positive action
Warning	`btn btn-warning`	Indicates caution should be taken with this action
Danger	`btn btn-danger`	Indicates a dangerous or potentially negative action
Inverse	`btn btn-inverse`	Alternate dark gray button, not tied to a semantic action or use
Link	`btn btn-link`	Deemphasize a button by making it look like a link while maintaining button behavior

Components defined in HTML, CSS and some jQuery

Then it was AngularJS time...

FrontSide



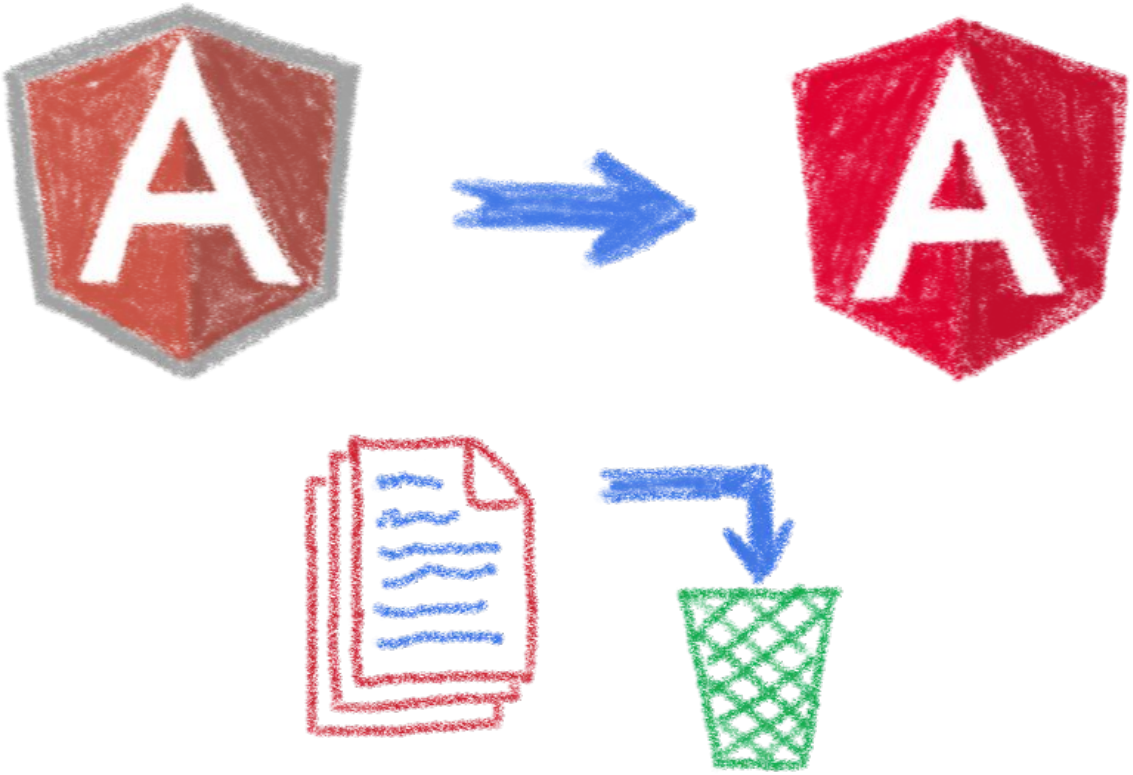
UI Bootstrap

Bootstrap components written in pure AngularJS by the AngularUI Team

Getting started

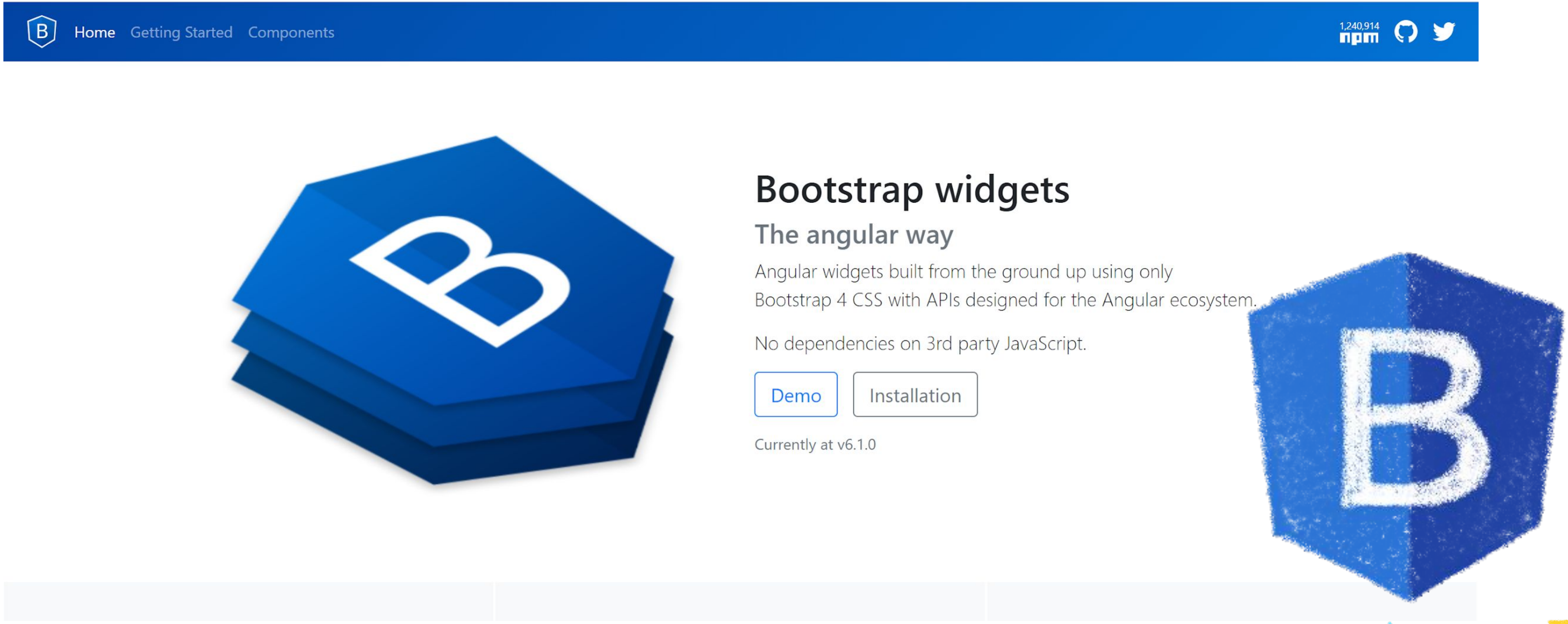
And new reference implementations were needed

But you know the sad story...



All UI Bootstrap based catalogs woke up with an obsolete implementation

Worry no more, let's do Angular!



Bootstrap widgets

The angular way

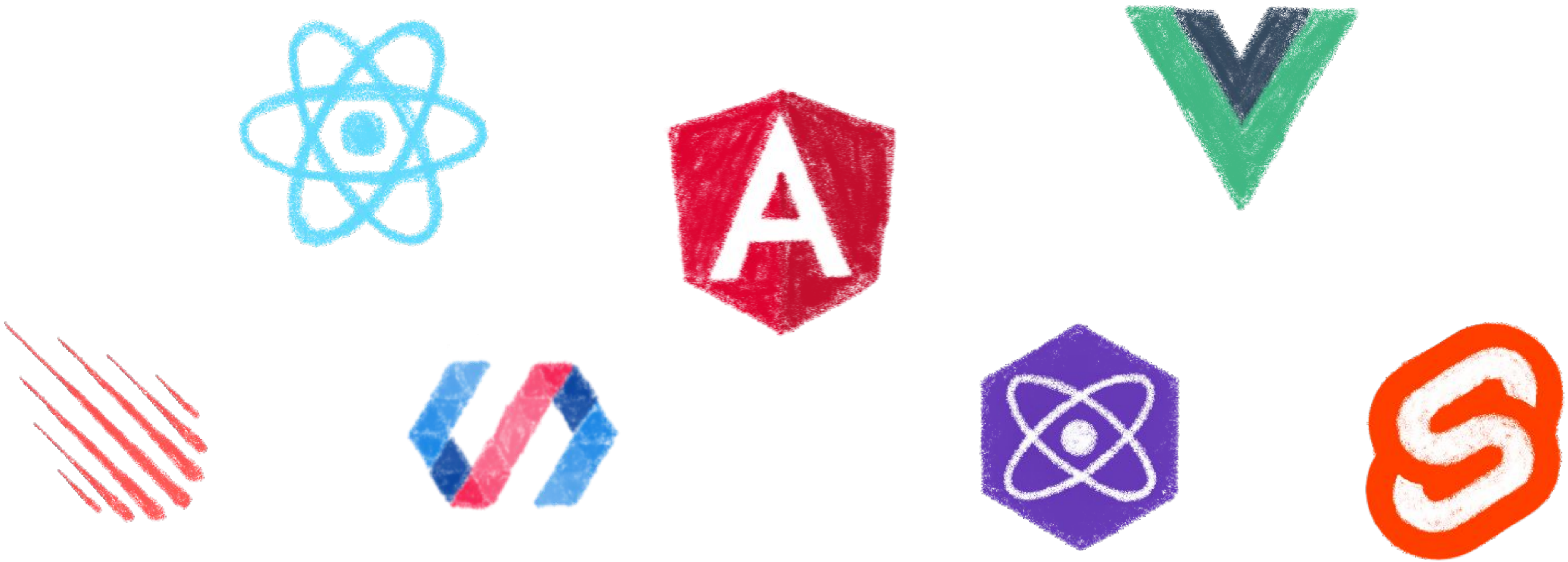
Angular widgets built from the ground up using only Bootstrap 4 CSS with APIs designed for the Angular ecosystem.

No dependencies on 3rd party JavaScript.

Currently at v6.1.0

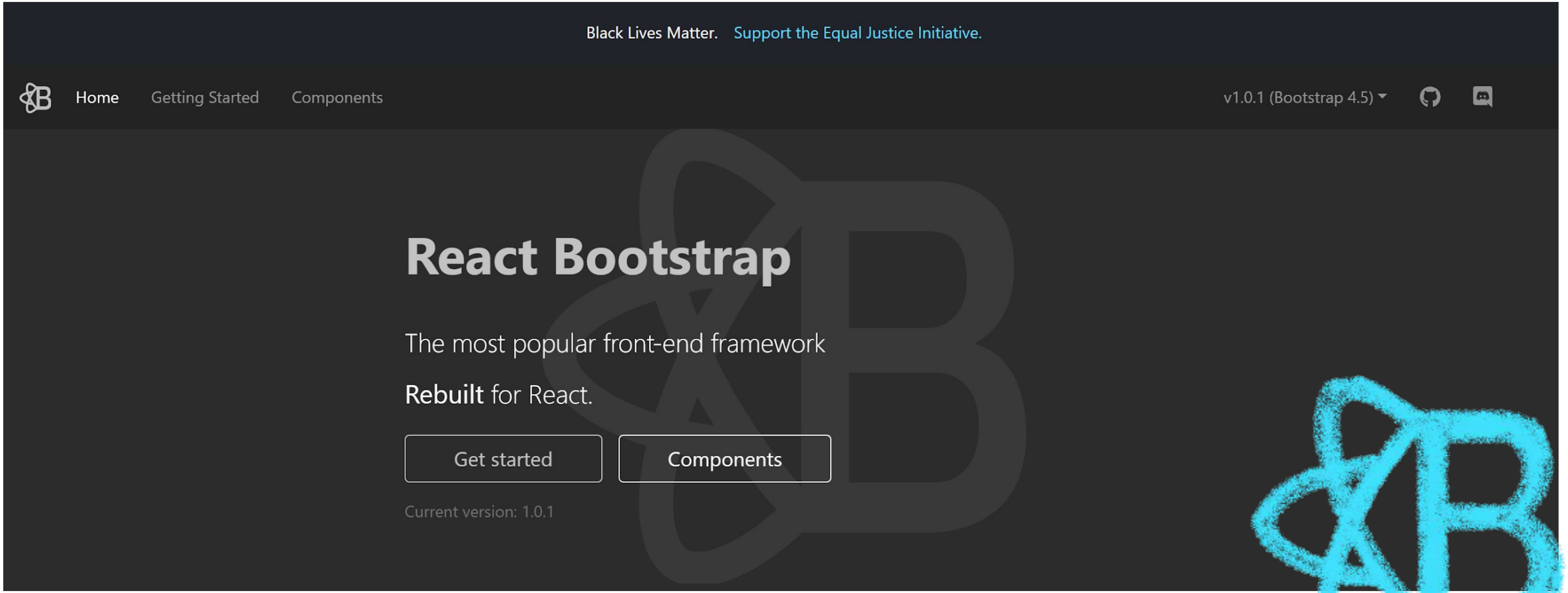
ng-bootstrap to the rescue

But times had changed...



In 2017 Angular is only one more in the clique

React is the new Big Thing™



Black Lives Matter. Support the Equal Justice Initiative.

React Bootstrap

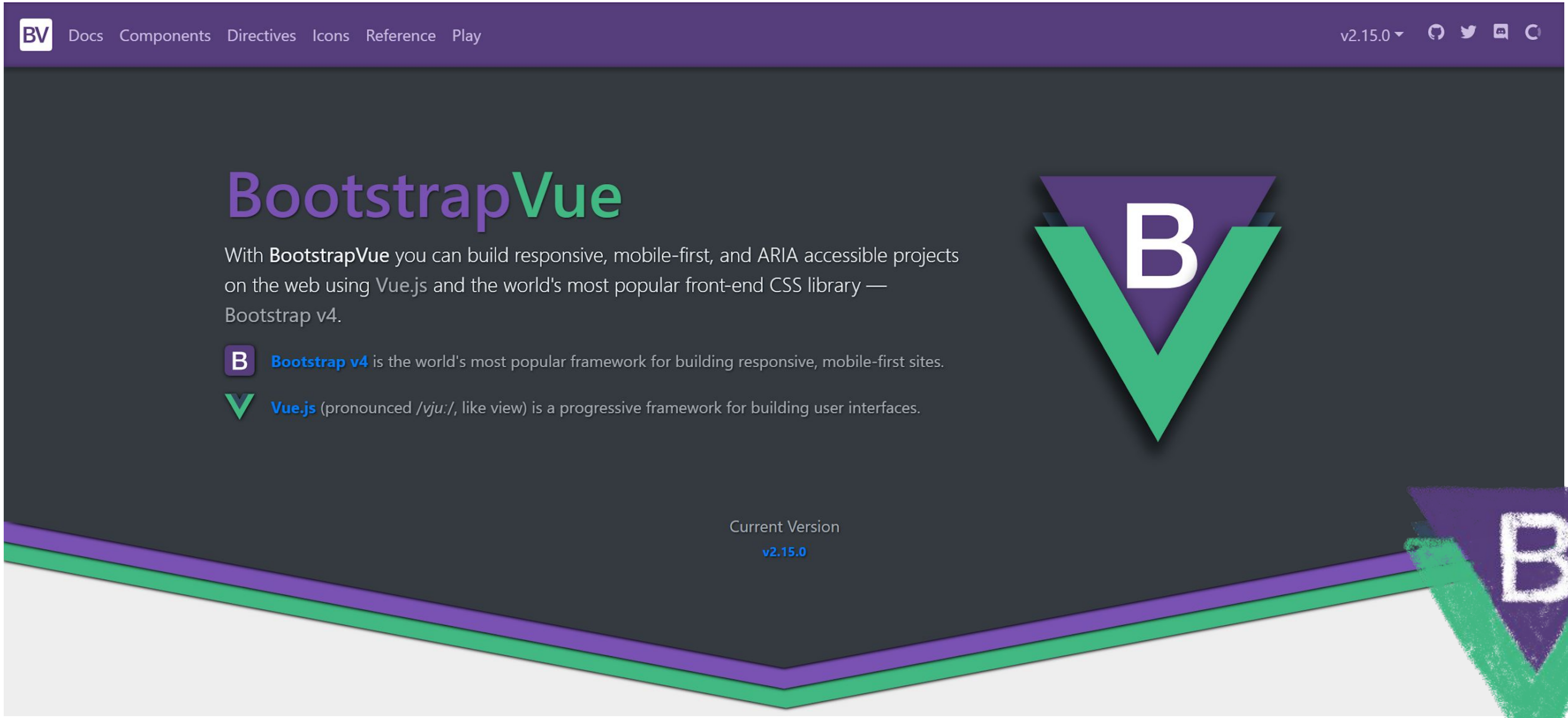
The most popular front-end framework

Rebuilt for React.

Current version: 1.0.1

So let's build React Bootstrap...

Wait, what about Vue?



BV Docs Components Directives Icons Reference Play v2.15.0

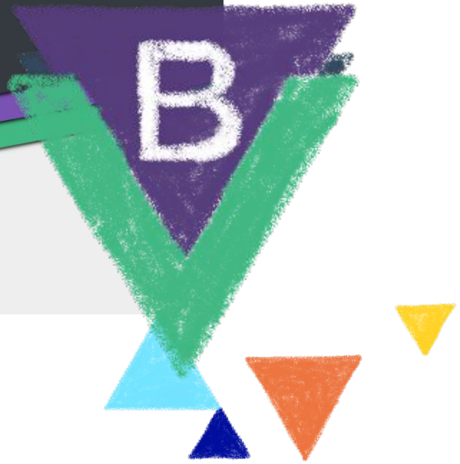
BootstrapVue

With BootstrapVue you can build responsive, mobile-first, and ARIA accessible projects on the web using Vue.js and the world's most popular front-end CSS library — Bootstrap v4.

B **Bootstrap v4** is the world's most popular framework for building responsive, mobile-first sites.

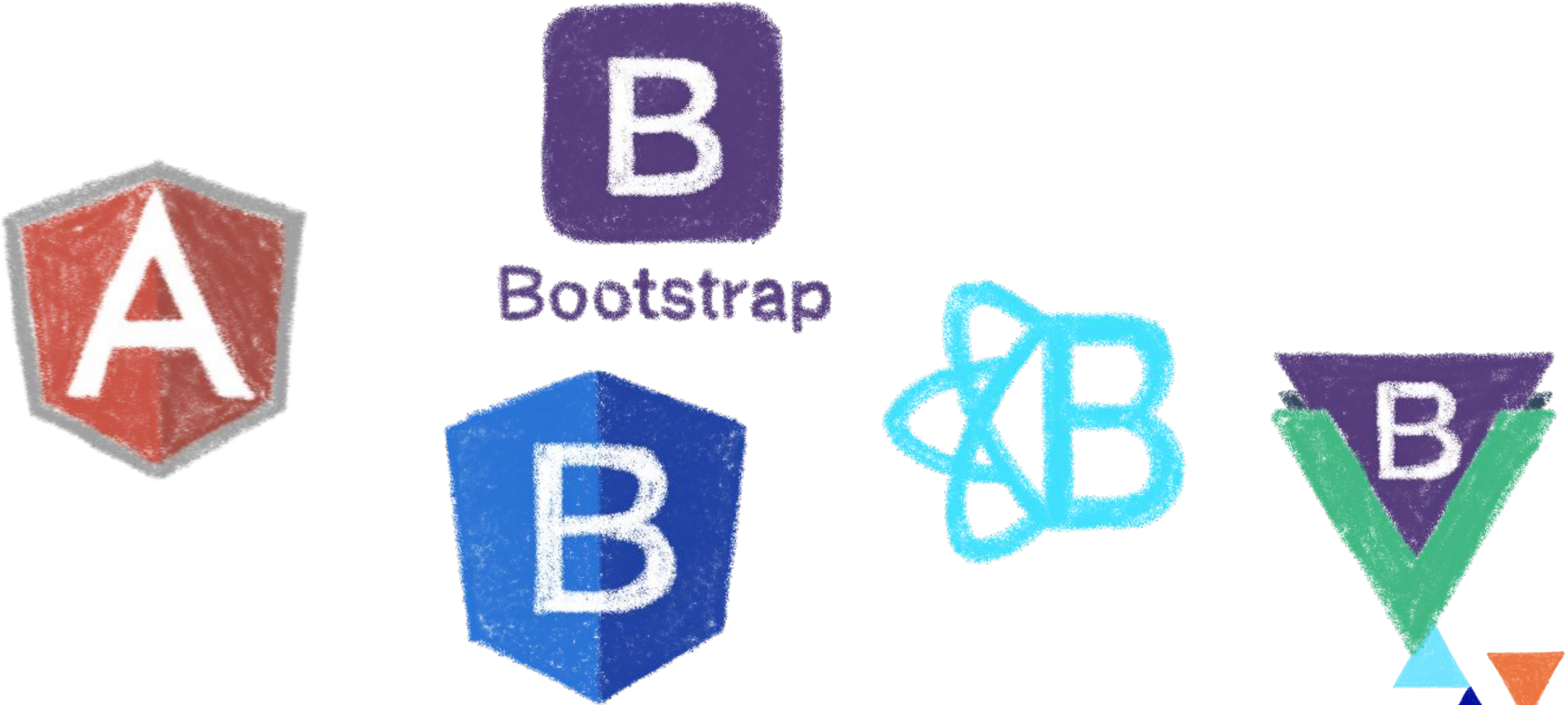
V **Vue.js** (pronounced /vjuː/, like view) is a progressive framework for building user interfaces.

Current Version
v2.15.0

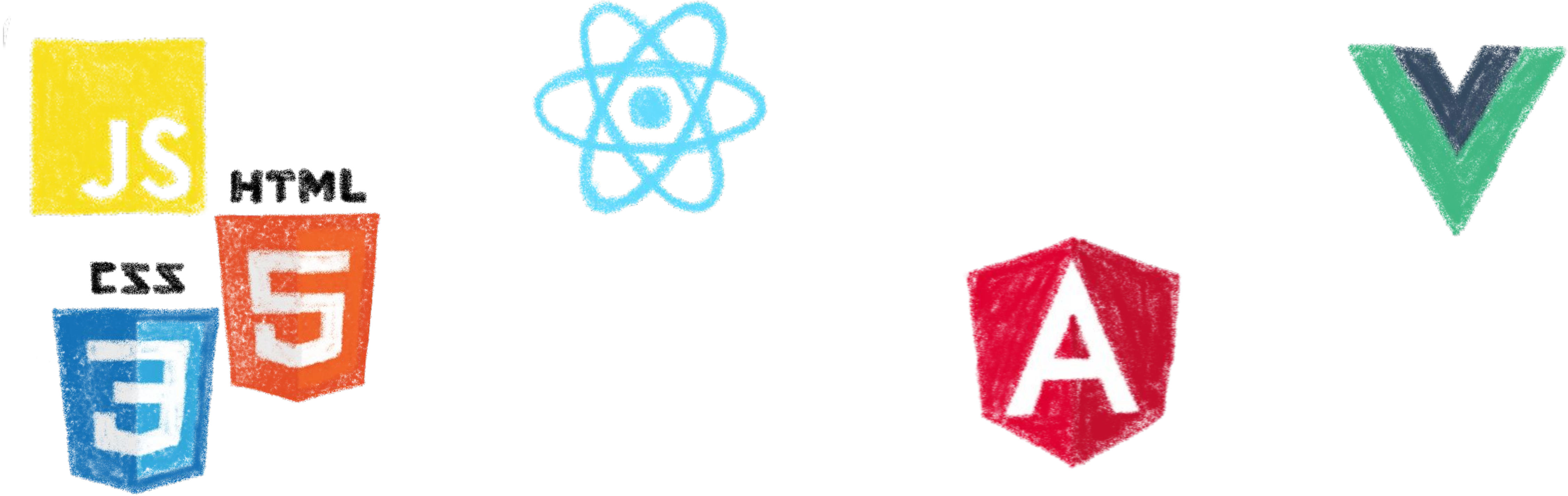


We also need BootstrapVue

OK, I think you see my point...

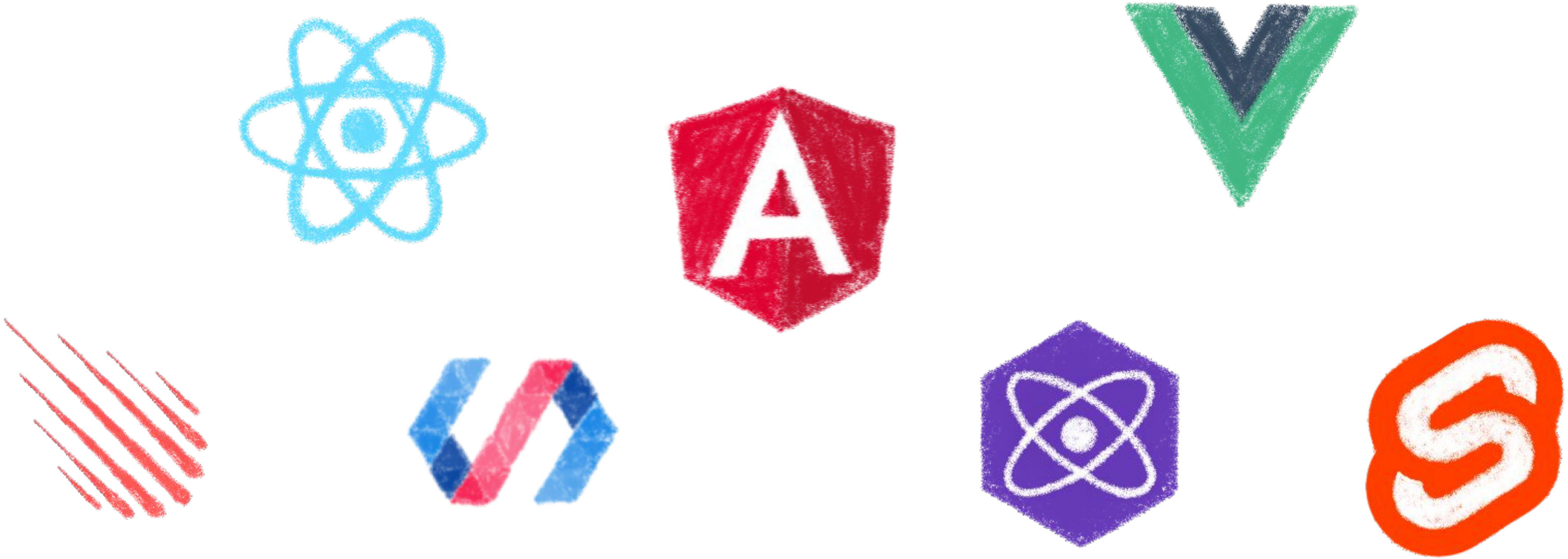


Most Design System do a choice



Either they choose a canonical implementation
or they ship and maintain several implementations

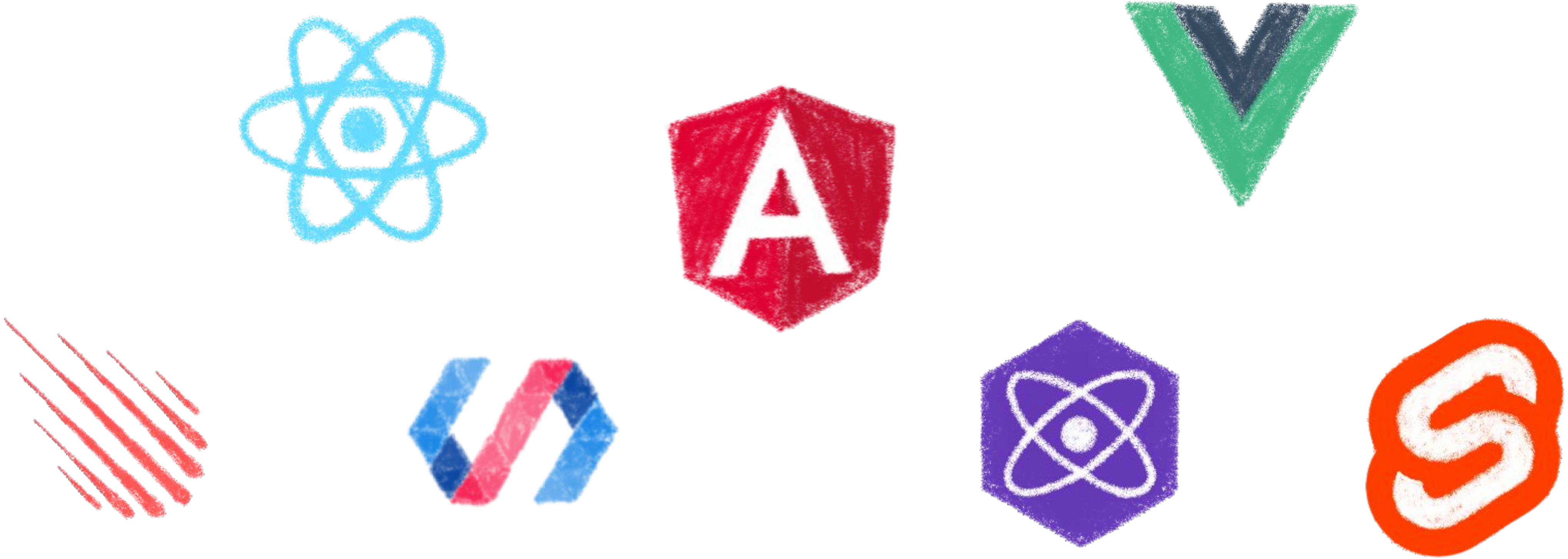
Both choices are problematic



Shipping only one implementation:

Web dev ecosystem changes quickly and almost nobody keeps the same framework for years...

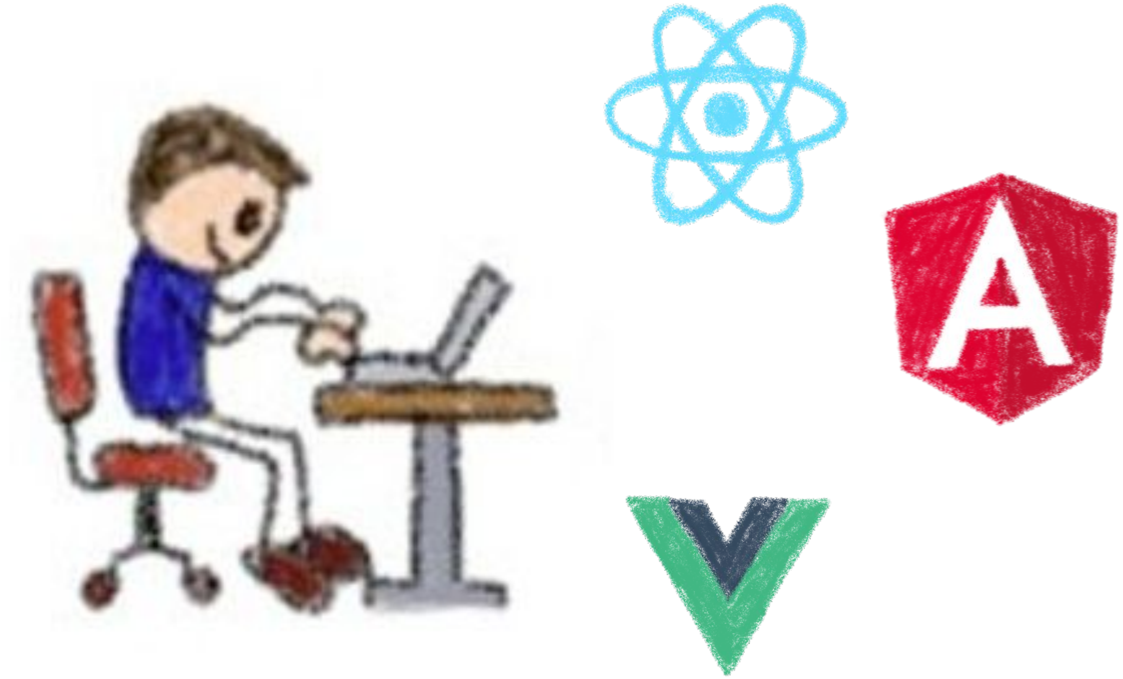
Both choices are problematic



Shipping several implementations:

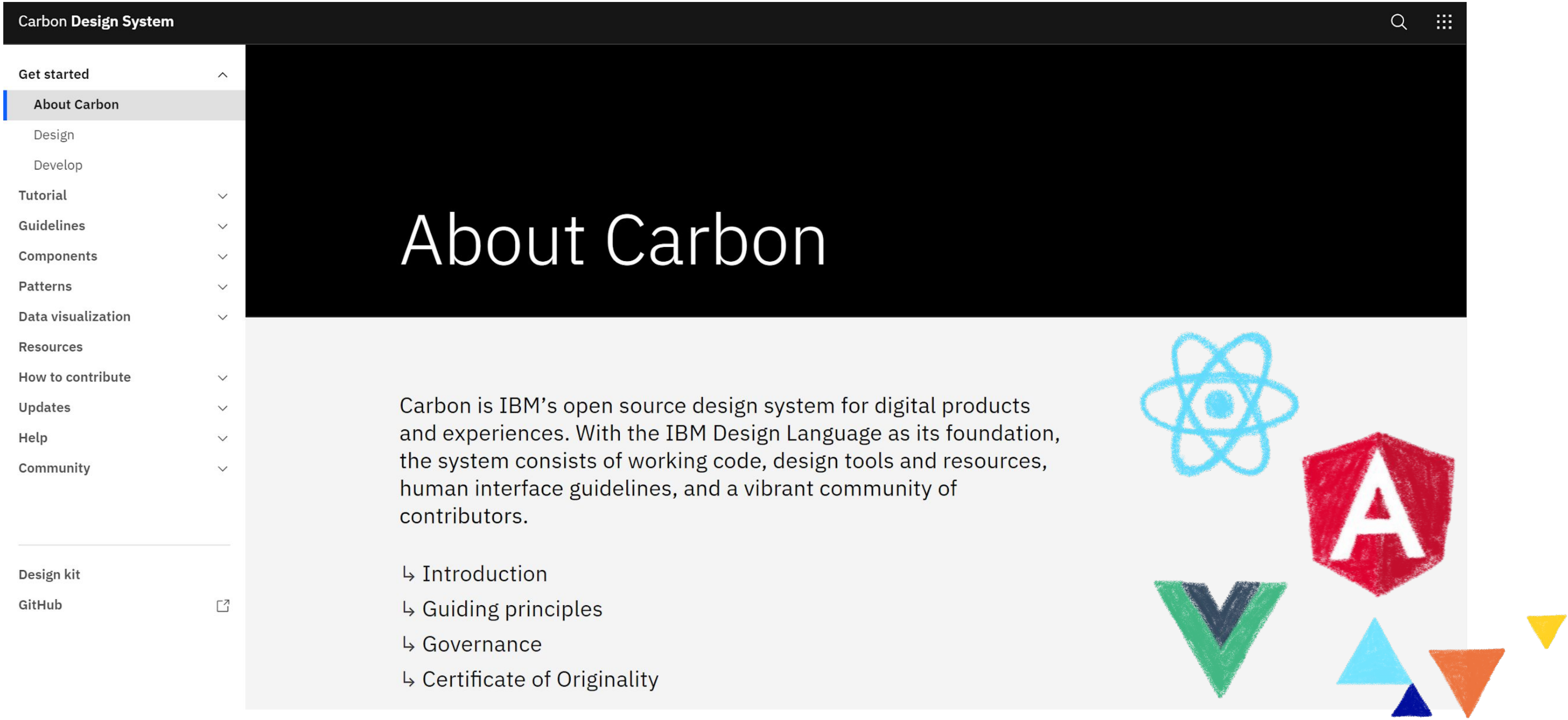
You need to maintain all the implementation…
and you still miss some others

Incomplete catalogs are problematic



People will need to recode the components in their chosen framework…
Coherence is not guaranteed!!!

Example: Carbon Design System



Carbon Design System

About Carbon

Carbon is IBM’s open source design system for digital products and experiences. With the IBM Design Language as its foundation, the system consists of working code, design tools and resources, human interface guidelines, and a vibrant community of contributors.

↳ Introduction

↳ Guiding principles

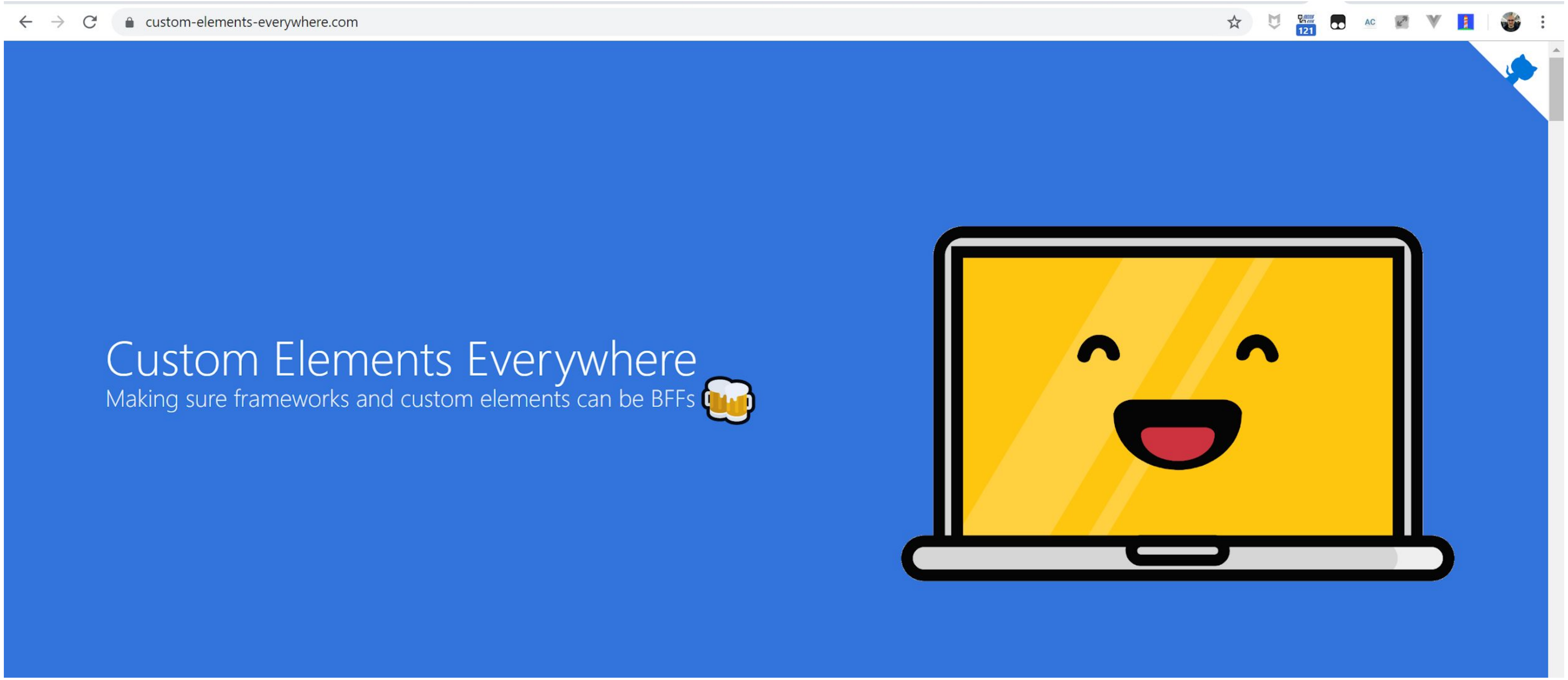
↳ Governance

↳ Certificate of Originality

Web Components & Design Systems

A match made in heaven

Compatibility is on Web Components side



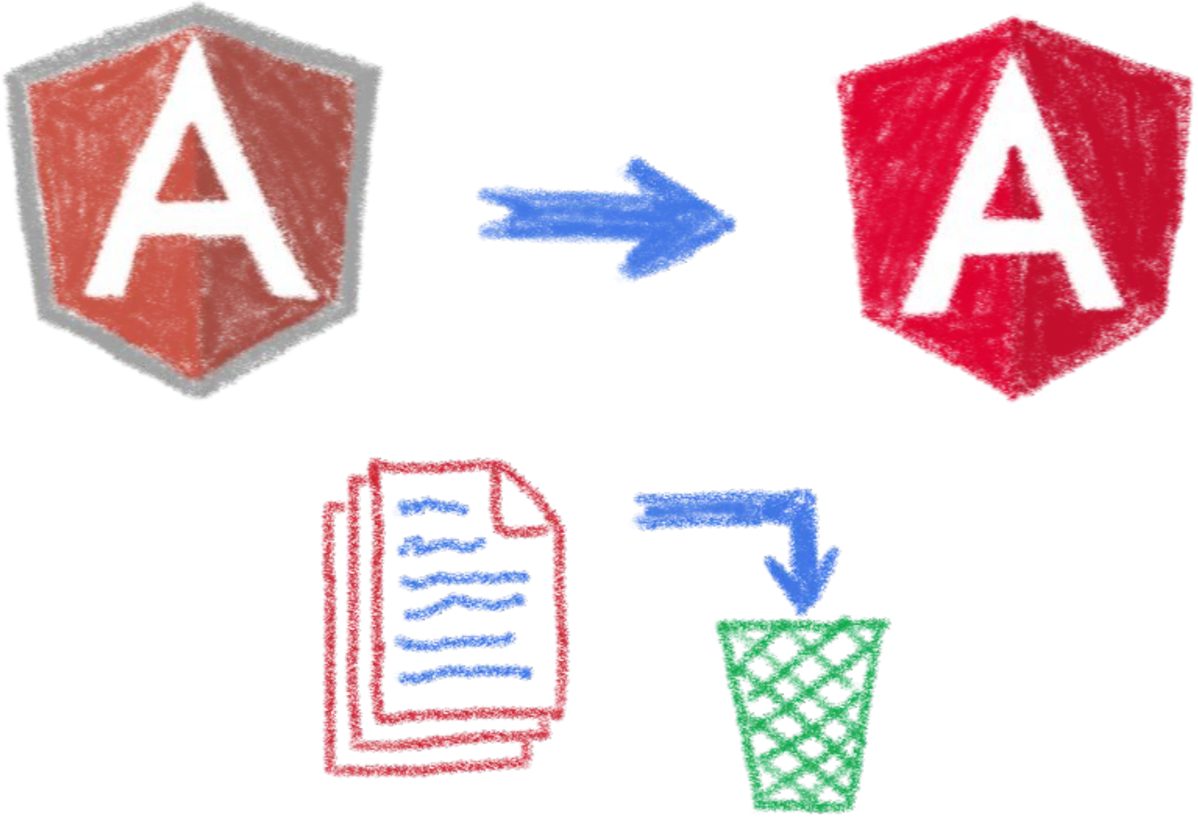
custom-elements-everywhere.com

Custom Elements Everywhere

Making sure frameworks and custom elements can be BFFs 🍻

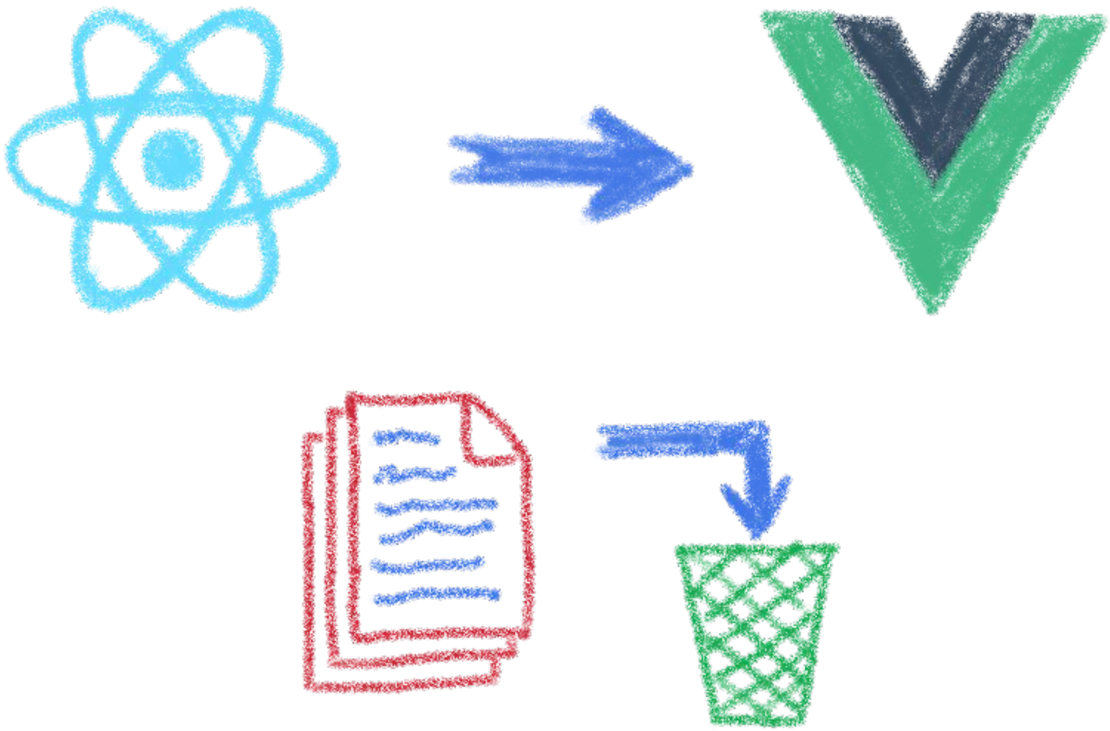
Web Components everywhere, baby!

Do you remember AngularJS?



And all the code put in the trash bin when Angular arrived...

The pain of switching frameworks?

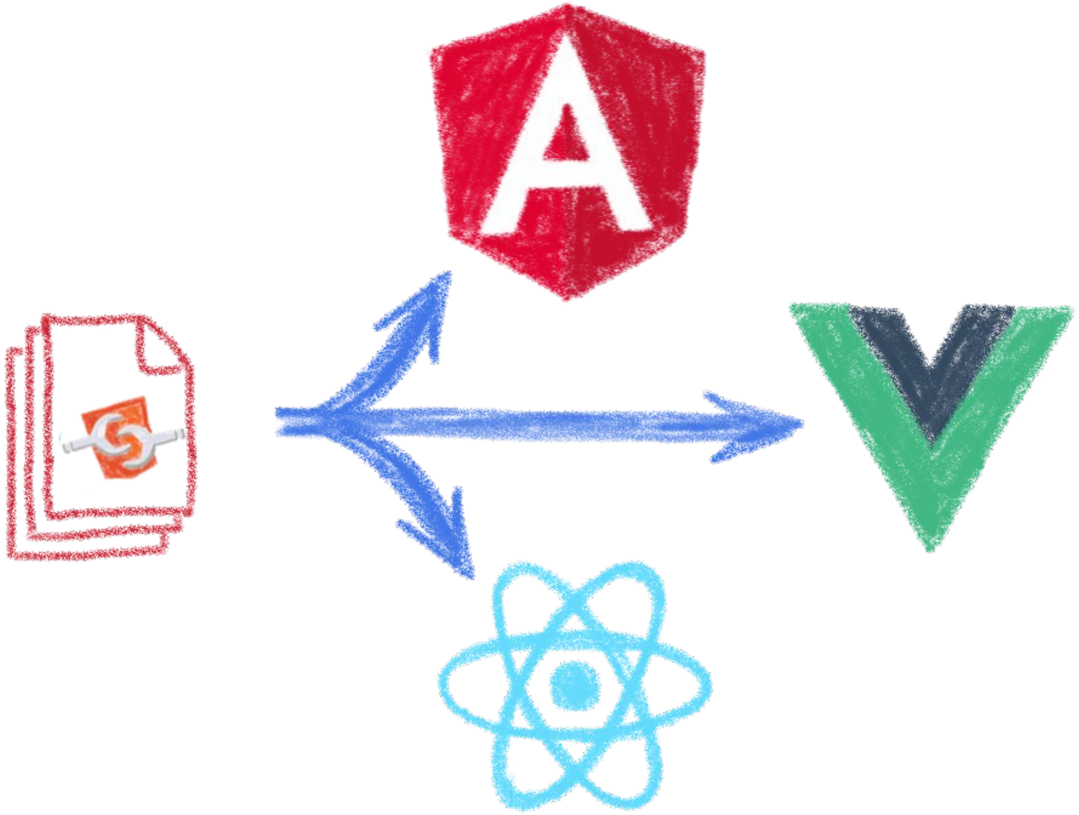


Rewriting once again your code...

The impossibility of sharing UI code?

Between apps written with different frameworks

Web Components change that

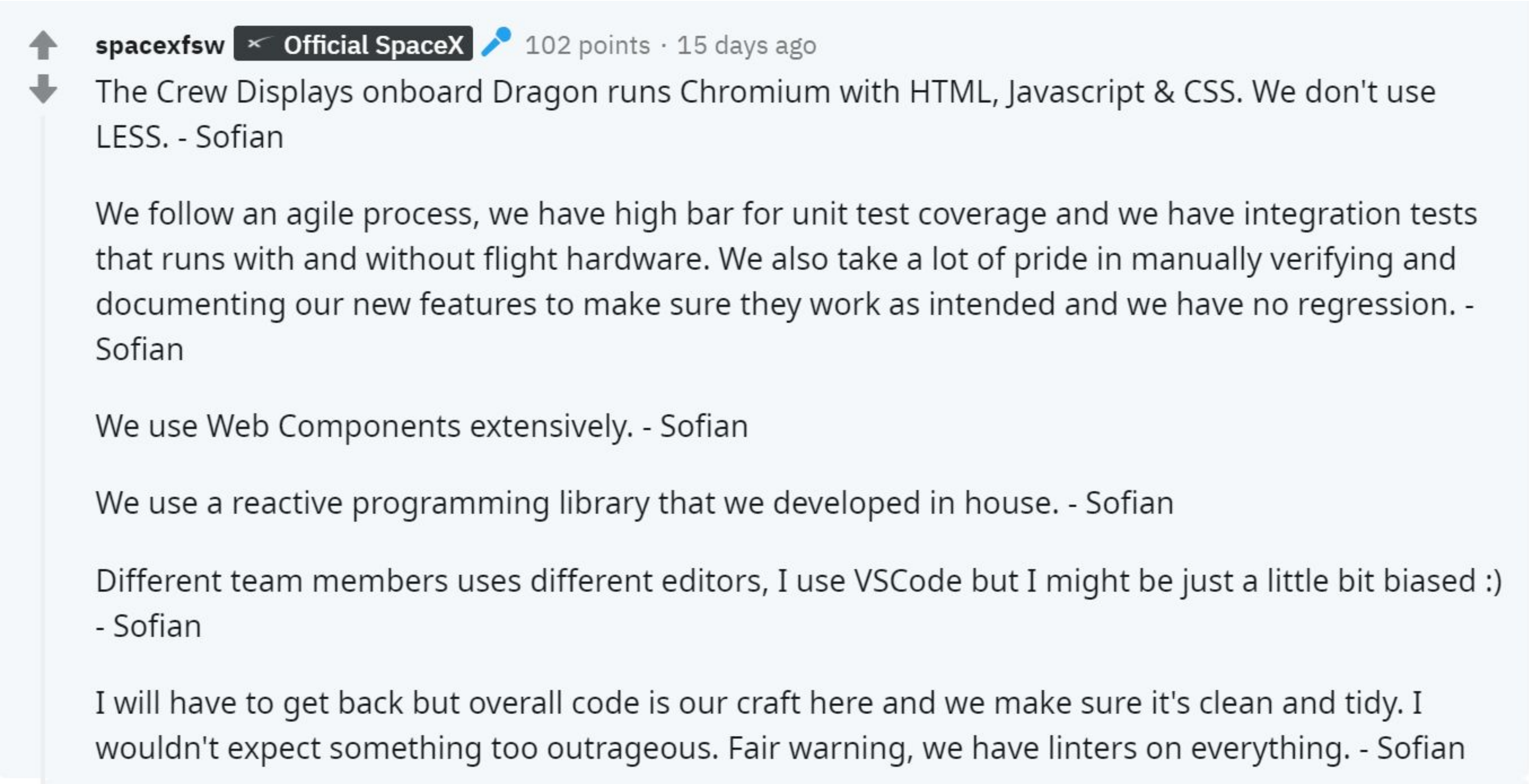


In a clean and standard way

They are the interoperable alternative

Any framework… or no framework at all

They are truly everywhere 🚀



spacexfsw Official SpaceX 102 points · 15 days ago

The Crew Displays onboard Dragon runs Chromium with HTML, Javascript & CSS. We don't use LESS. - Sofian

We follow an agile process, we have high bar for unit test coverage and we have integration tests that runs with and without flight hardware. We also take a lot of pride in manually verifying and documenting our new features to make sure they work as intended and we have no regression. - Sofian

We use Web Components extensively. - Sofian

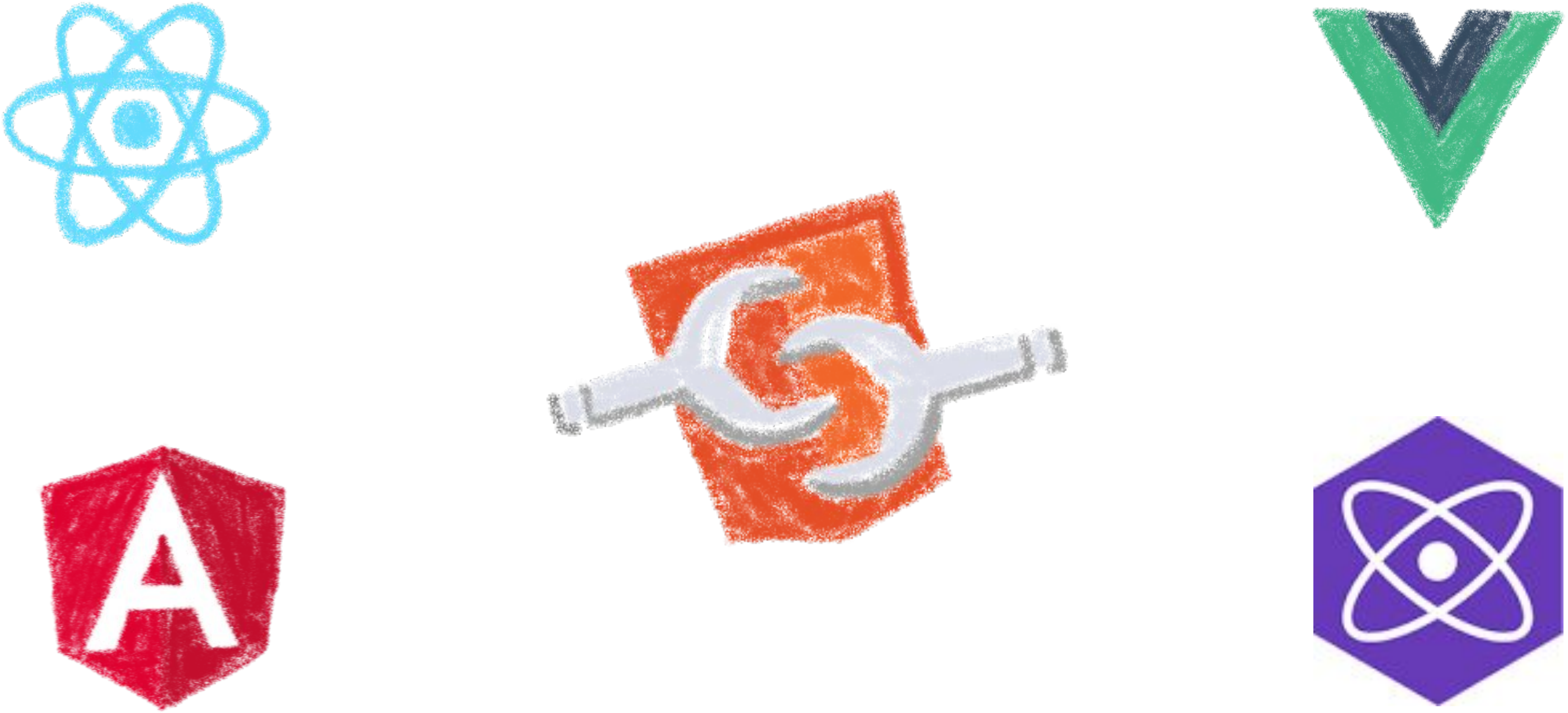
We use a reactive programming library that we developed in house. - Sofian

Different team members uses different editors, I use VSCode but I might be just a little bit biased :) - Sofian

I will have to get back but overall code is our craft here and we make sure it's clean and tidy. I wouldn't expect something too outrageous. Fair warning, we have linters on everything. - Sofian

🚀 Even in the spaaaaaaaace 🚀

You can have a single implementation



And it simply works everywhere

When you need interoperability

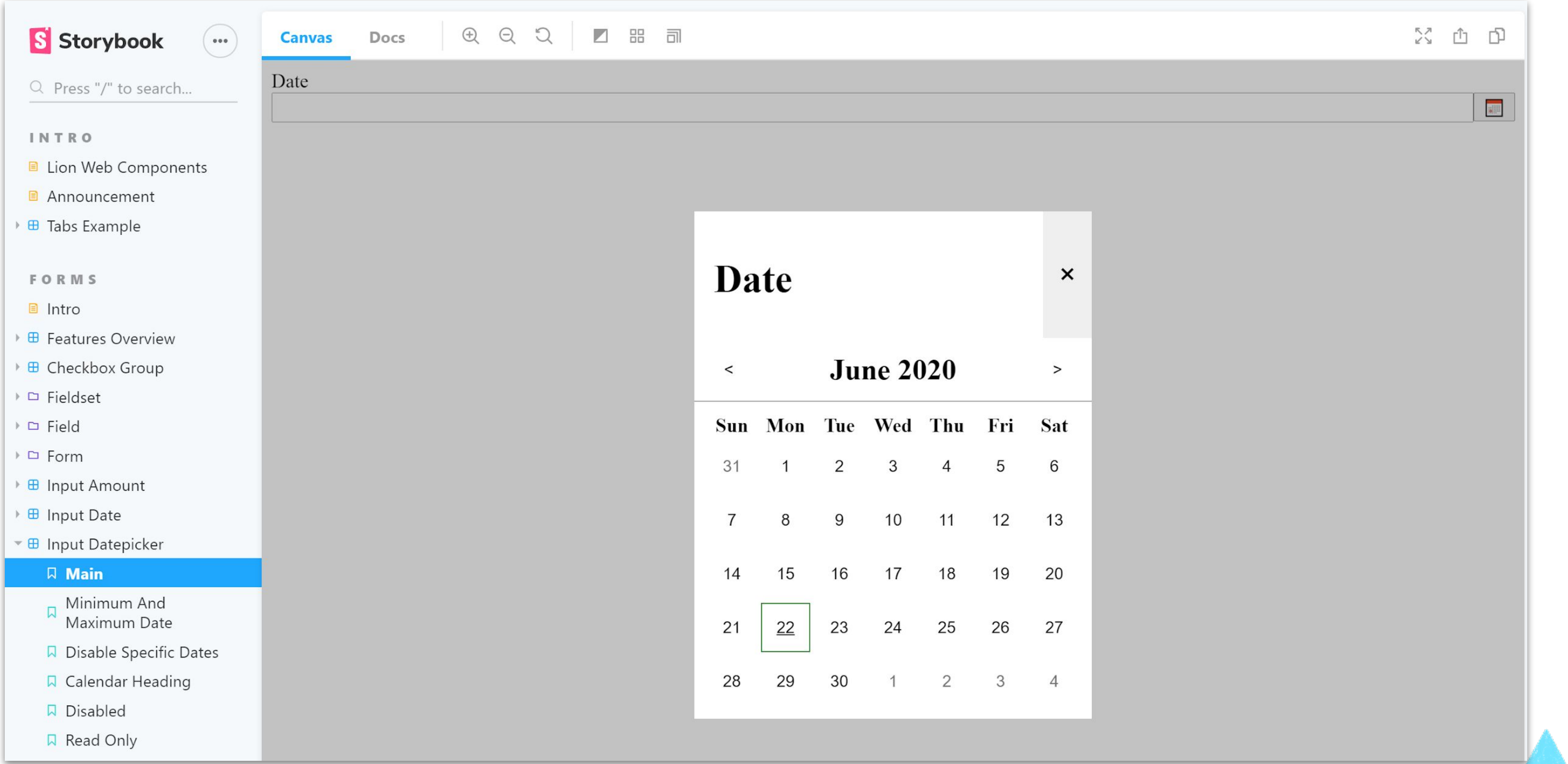
Nothing beats the standard

But how to do it?

Designing, developing and managing a catalog of Web Components

Learning from the best

FrontSide



Date

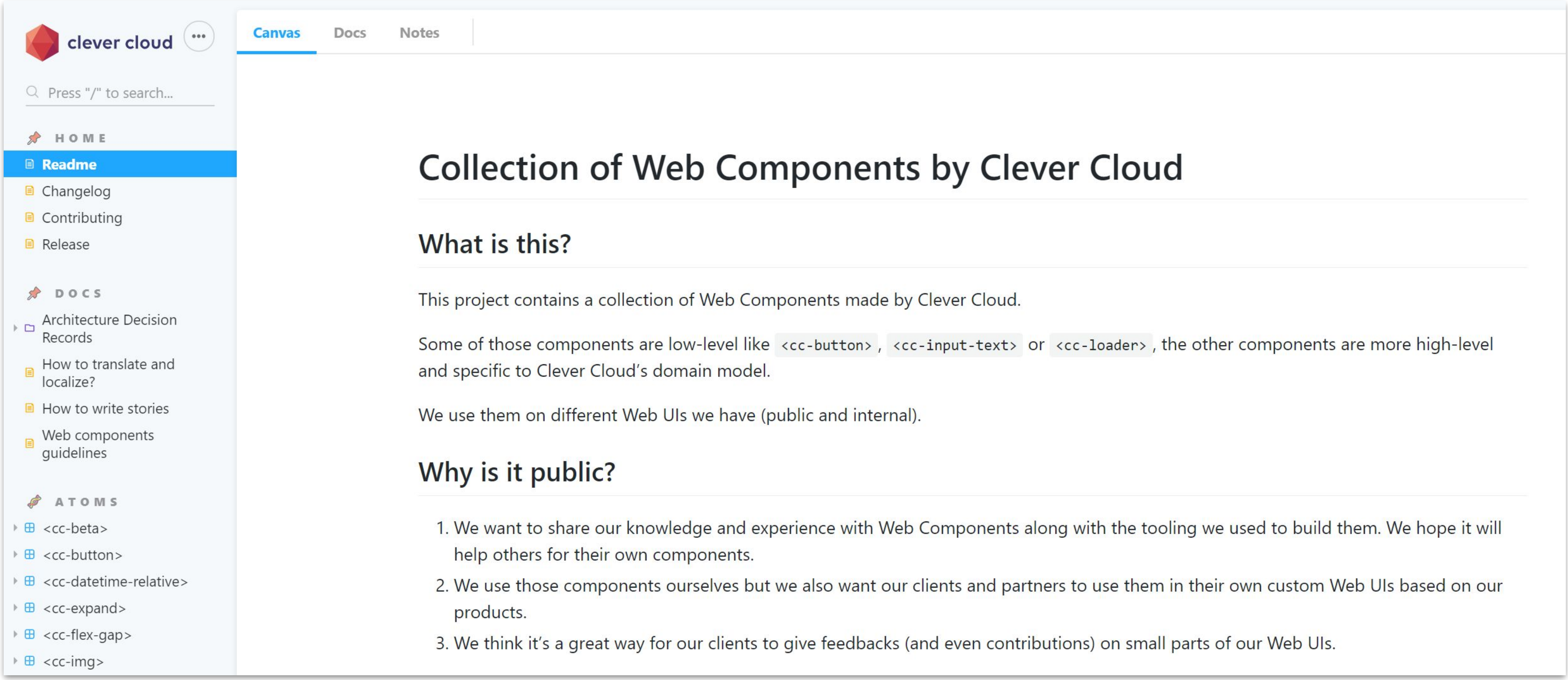
Date ×

< June 2020 >

Sun	Mon	Tue	Wed	Thu	Fri	Sat
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	1	2	3	4

https://lion-web-components.netlify.app/

Learning from the best



Collection of Web Components by Clever Cloud

What is this?

This project contains a collection of Web Components made by Clever Cloud.

Some of those components are low-level like `<cc-button>`, `<cc-input-text>` or `<cc-loader>`, the other components are more high-level and specific to Clever Cloud's domain model.

We use them on different Web UIs we have (public and internal).

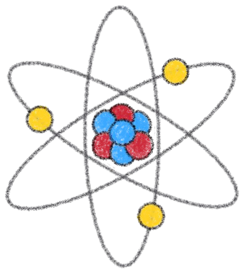
Why is it public?

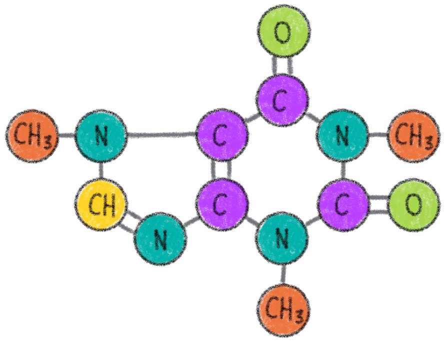
1. We want to share our knowledge and experience with Web Components along with the tooling we used to build them. We hope it will help others for their own components.
2. We use those components ourselves but we also want our clients and partners to use them in their own custom Web UIs based on our products.
3. We think it's a great way for our clients to give feedbacks (and even contributions) on small parts of our Web UIs.

https://github.com/CleverCloud/clever-components

What kind of components?

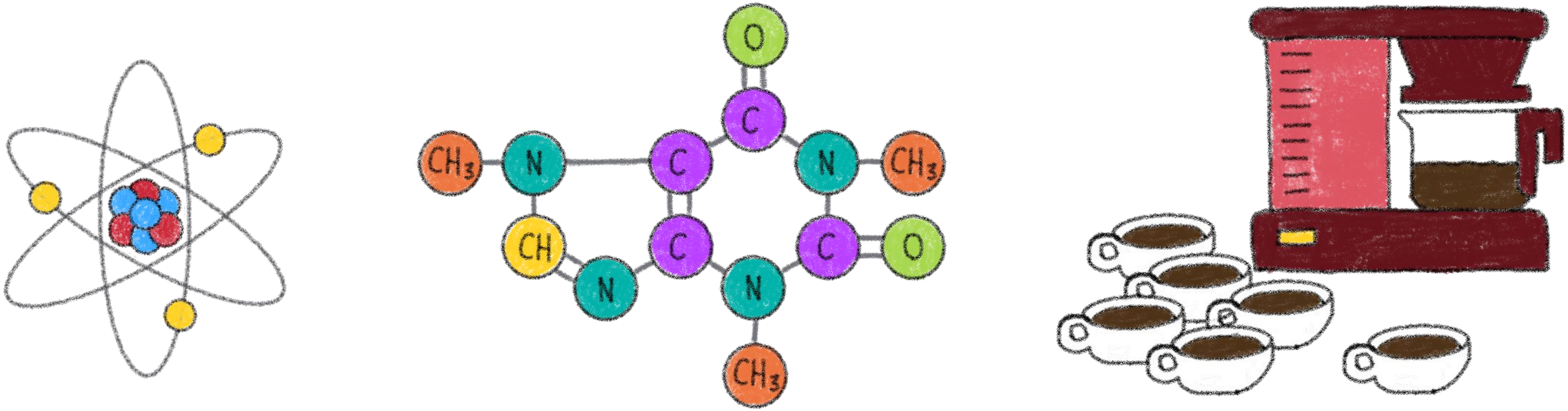
From little atomic blocs to big smart components, and everything in between





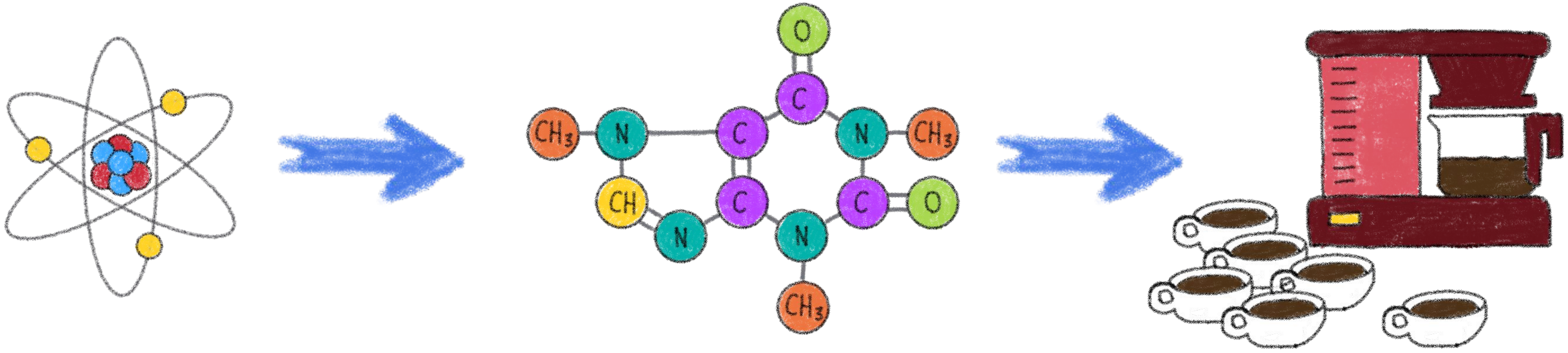
A matter of size and complexity

FrontSide



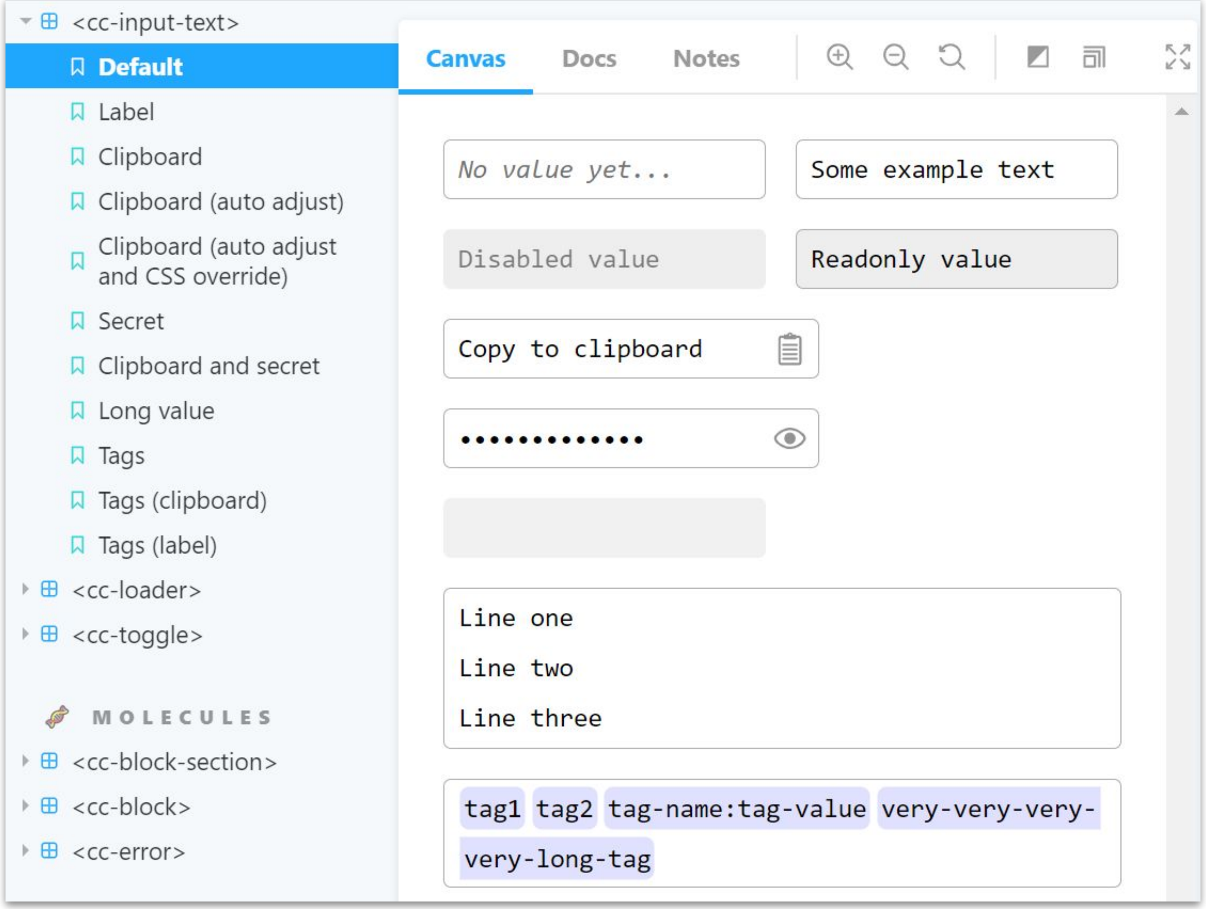
What kind(s) of components you want to build

Build from the bottom and go up



Eat your own dog food

And how to choose the atoms?



No value yet...

Some example text

Disabled value

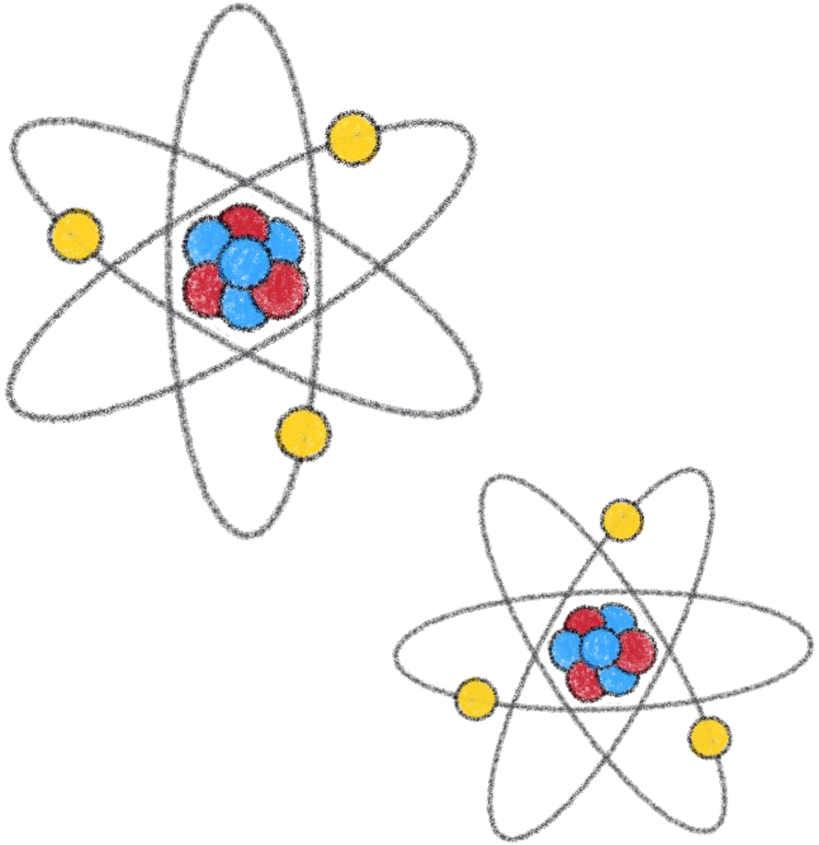
Readonly value

Copy to clipboard

••••••••••••••

Line one
Line two
Line three

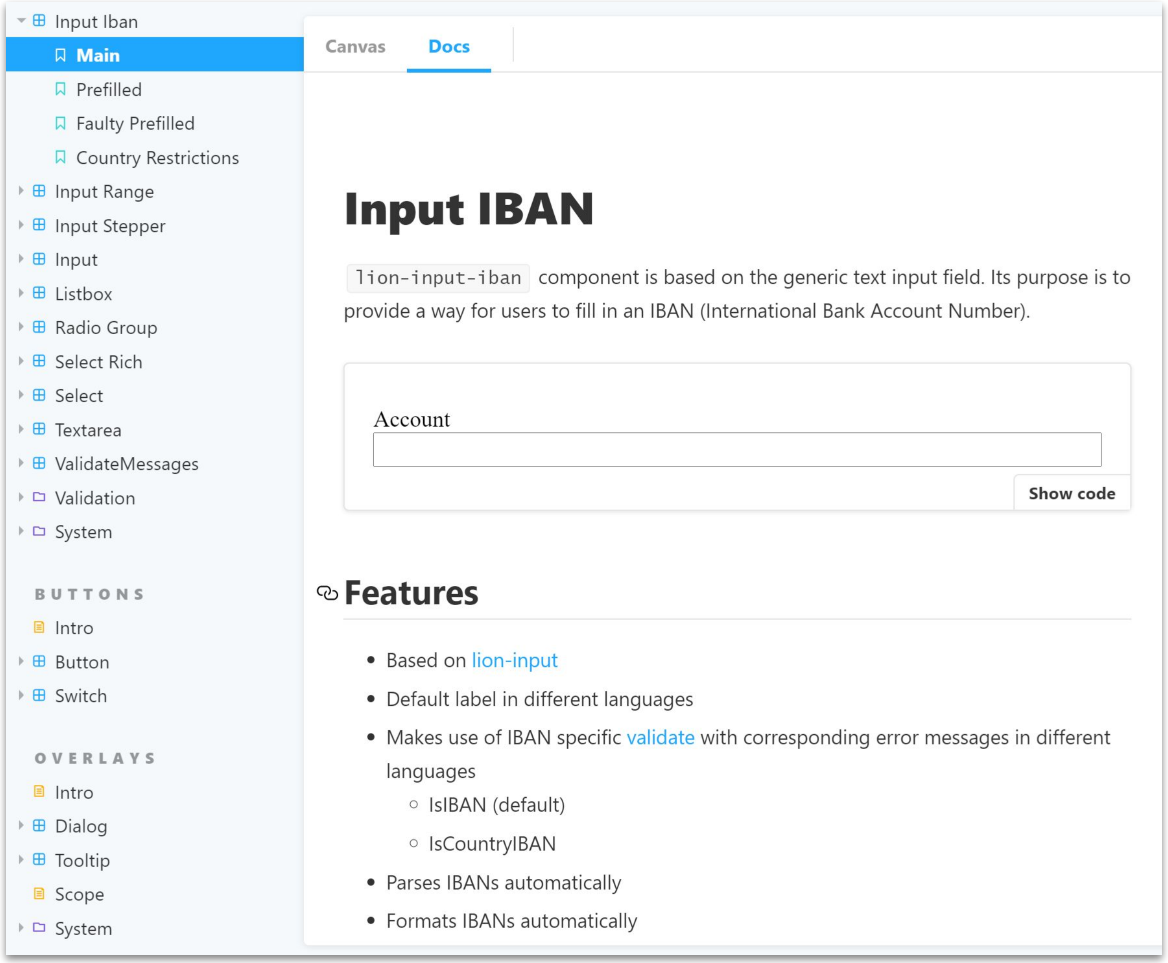
tag1 tag2 tag-name:tag-value very-very-very-very-long-tag



Flexibility and configurability are key

And how to choose the atoms?

FrontSide

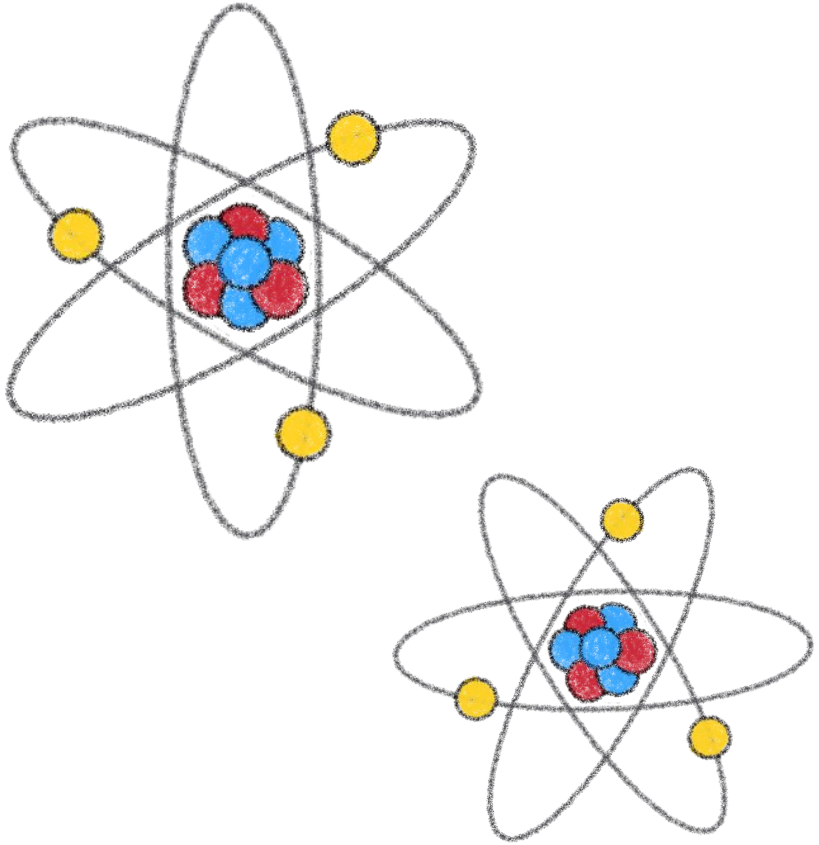


Input IBAN

`lion-input-iban` component is based on the generic text input field. Its purpose is to provide a way for users to fill in an IBAN (International Bank Account Number).

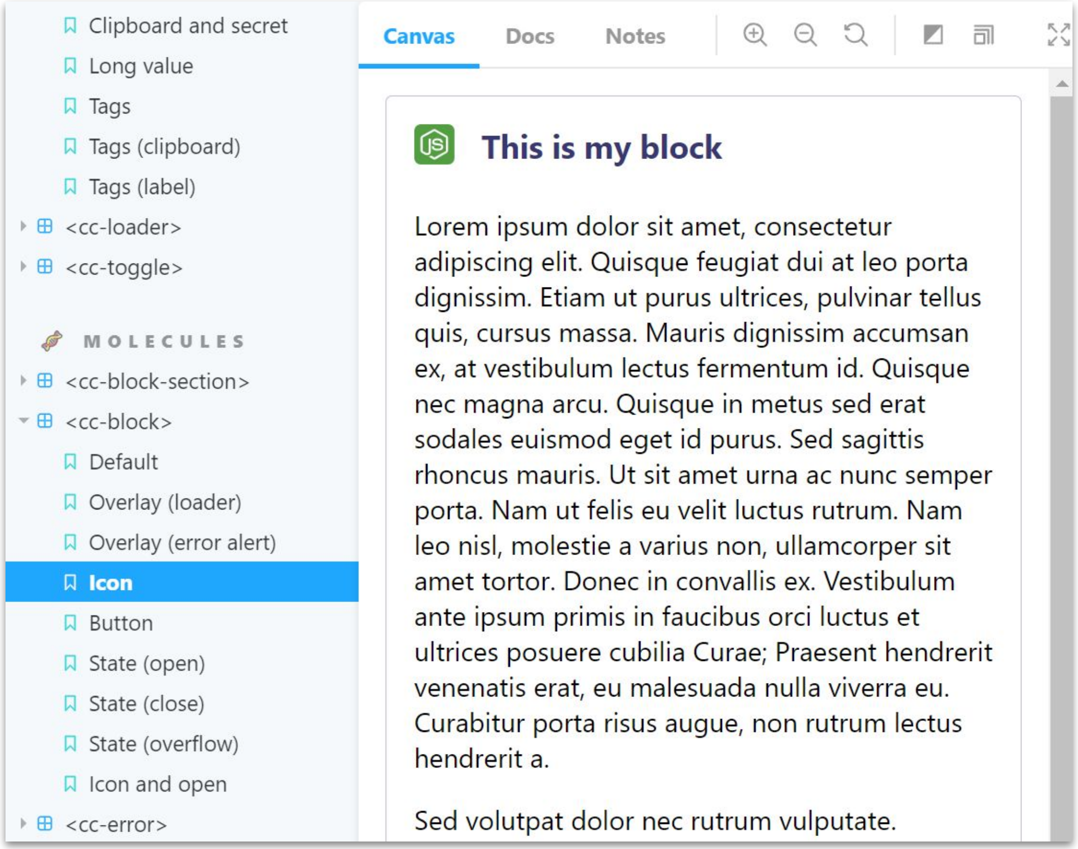
Features

- Based on lion-input
- Default label in different languages
- Makes use of IBAN specific validate with corresponding error messages in different languages
  - IsIBAN (default)
  - IsCountryIBAN
- Parses IBANs automatically
- Formats IBANs automatically



Encode often used patterns

And what about the molecules?



- Clipboard and secret
- Long value
- Tags
- Tags (clipboard)
- Tags (label)
- <cc-loader>
- <cc-toggle>

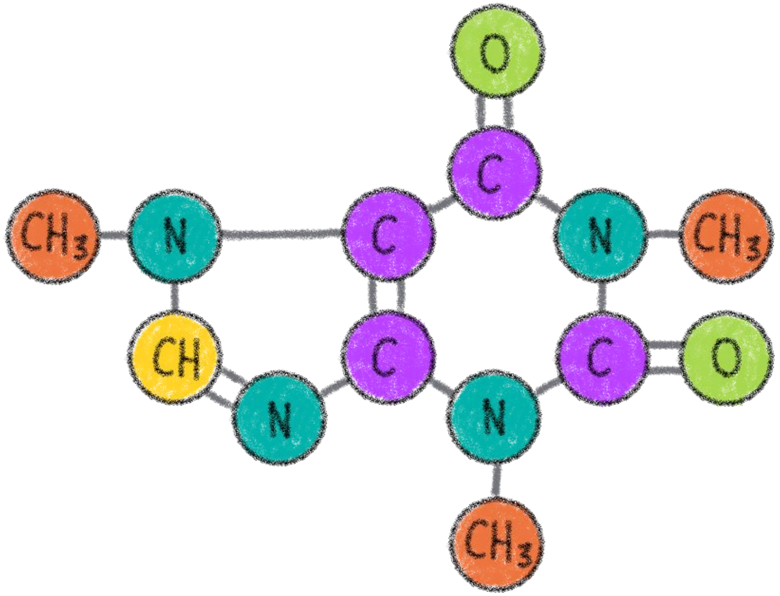
MOLECULES

- <cc-block-section>
- <cc-block>
  - Default
  - Overlay (loader)
  - Overlay (error alert)
  - Icon
  - Button
  - State (open)
  - State (close)
  - State (overflow)
  - Icon and open
- <cc-error>

This is my block

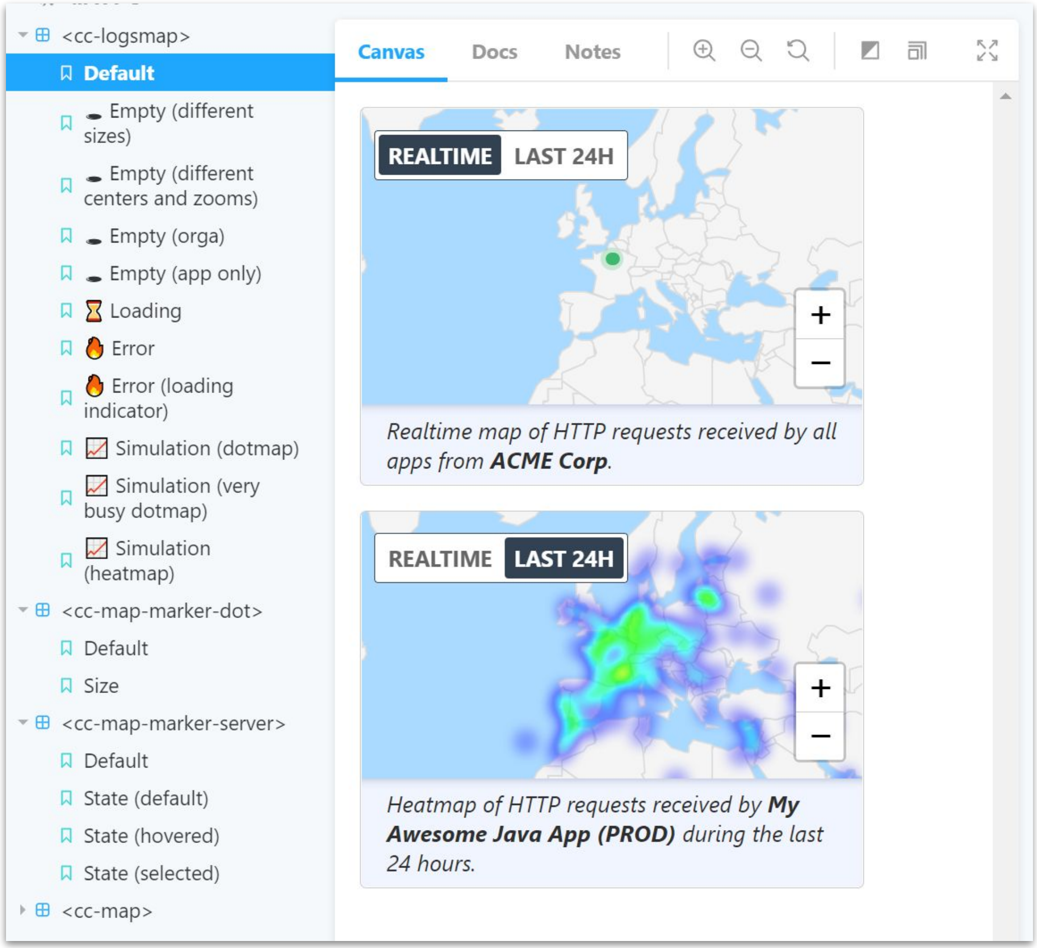
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque feugiat dui at leo porta dignissim. Etiam ut purus ultrices, pulvinar tellus quis, cursus massa. Mauris dignissim accumsan ex, at vestibulum lectus fermentum id. Quisque nec magna arcu. Quisque in metus sed erat sodales euismod eget id purus. Sed sagittis rhoncus mauris. Ut sit amet urna ac nunc semper porta. Nam ut felis eu velit luctus rutrum. Nam leo nisl, molestie a varius non, ullamcorper sit amet tortor. Donec in convallis ex. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Praesent hendrerit venenatis erat, eu malesuada nulla viverra eu. Curabitur porta risus augue, non rutrum lectus hendrerit a.

Sed volutpat dolor nec rutrum vulputate.



Capitalize on your atoms
Keep the flexibility and configurability

Big smart business components



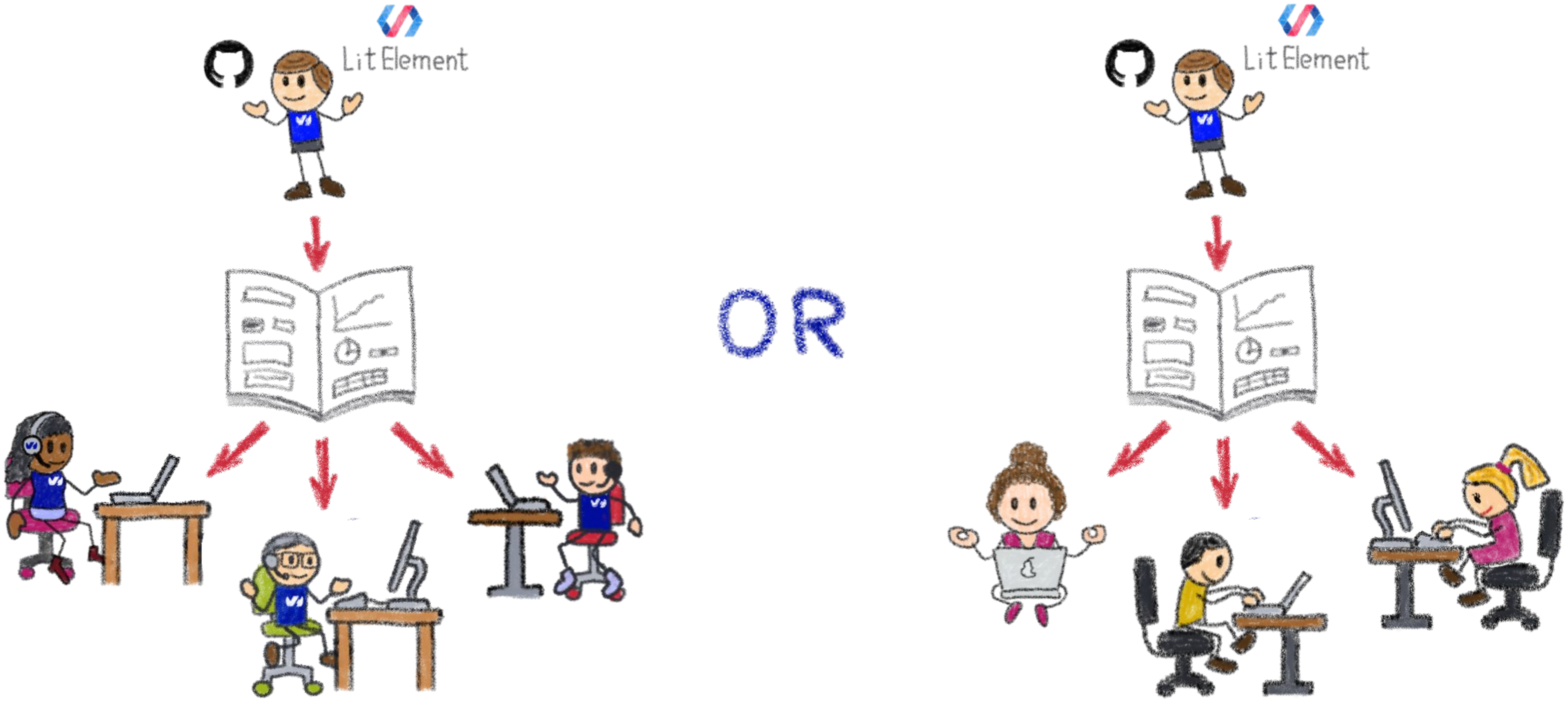
- <cc-logsmap>
  - Default
  - Empty (different sizes)
  - Empty (different centers and zooms)
  - Empty (orga)
  - Empty (app only)
  - ⌛ Loading
  - 🔥 Error
  - 🔥 Error (loading indicator)
  - 📈 Simulation (dotmap)
  - 📈 Simulation (very busy dotmap)
  - 📈 Simulation (heatmap)
- <cc-map-marker-dot>
  - Default
  - Size
- <cc-map-marker-server>
  - Default
  - State (default)
  - State (hovered)
  - State (selected)
- <cc-map>

*Realtime map of HTTP requests received by all apps from **ACME Corp**.*

*Heatmap of HTTP requests received by **My Awesome Java App (PROD)** during the last 24 hours.*

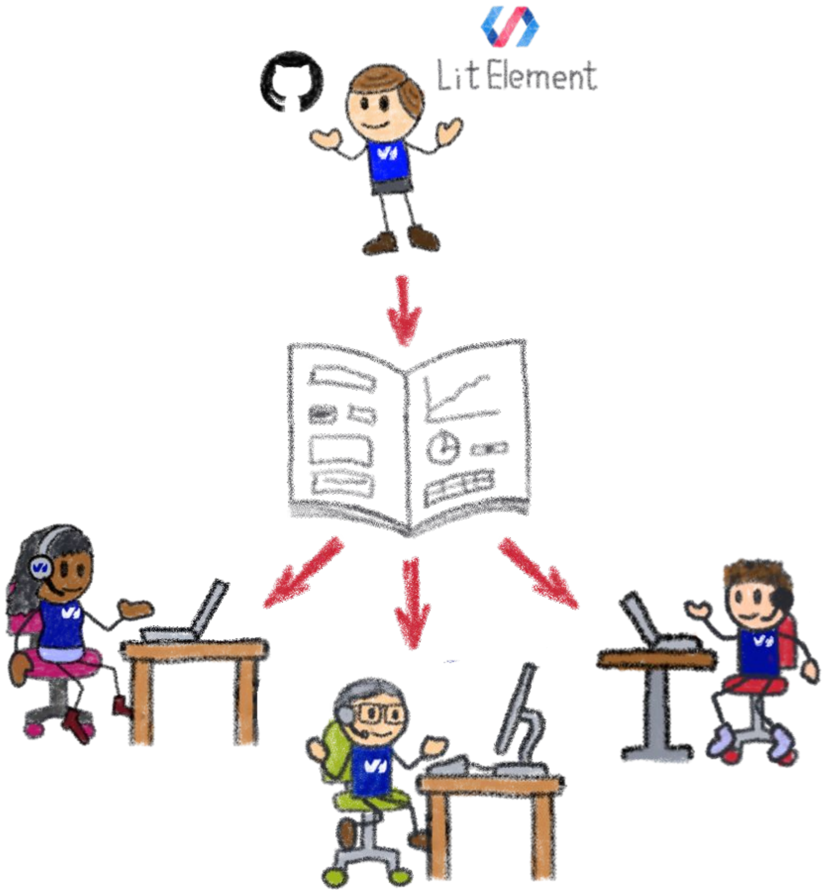
Encoding your business logic

Internal or external customers?



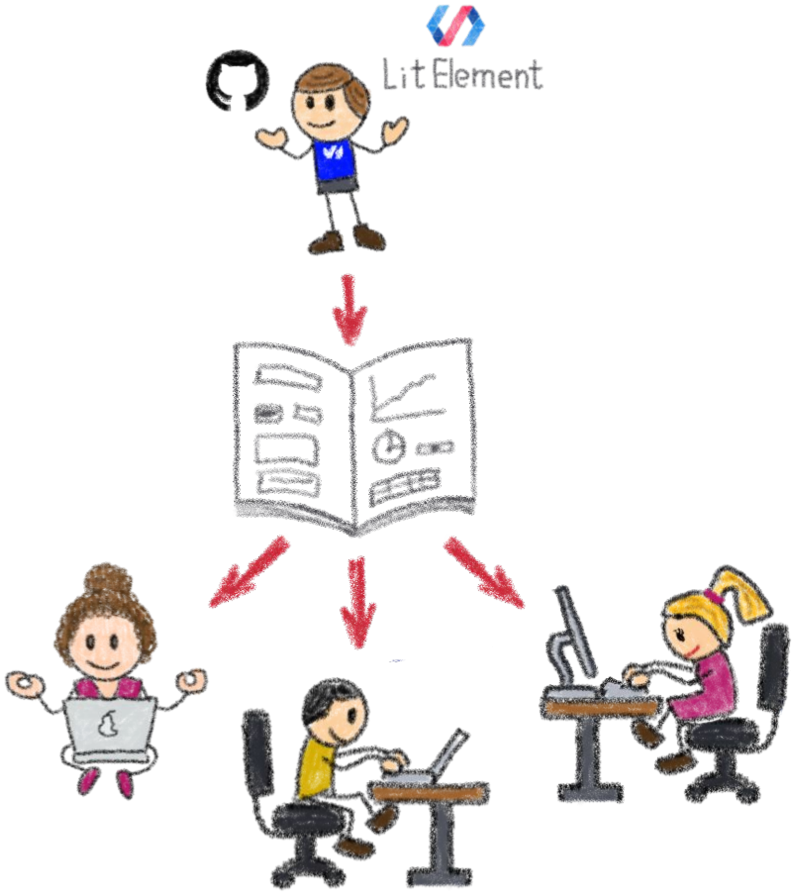
Who are your target users?

Internal customers need off-the-shelf components



A well defined and coherent look-and-feel

External customers need to be able to tweak

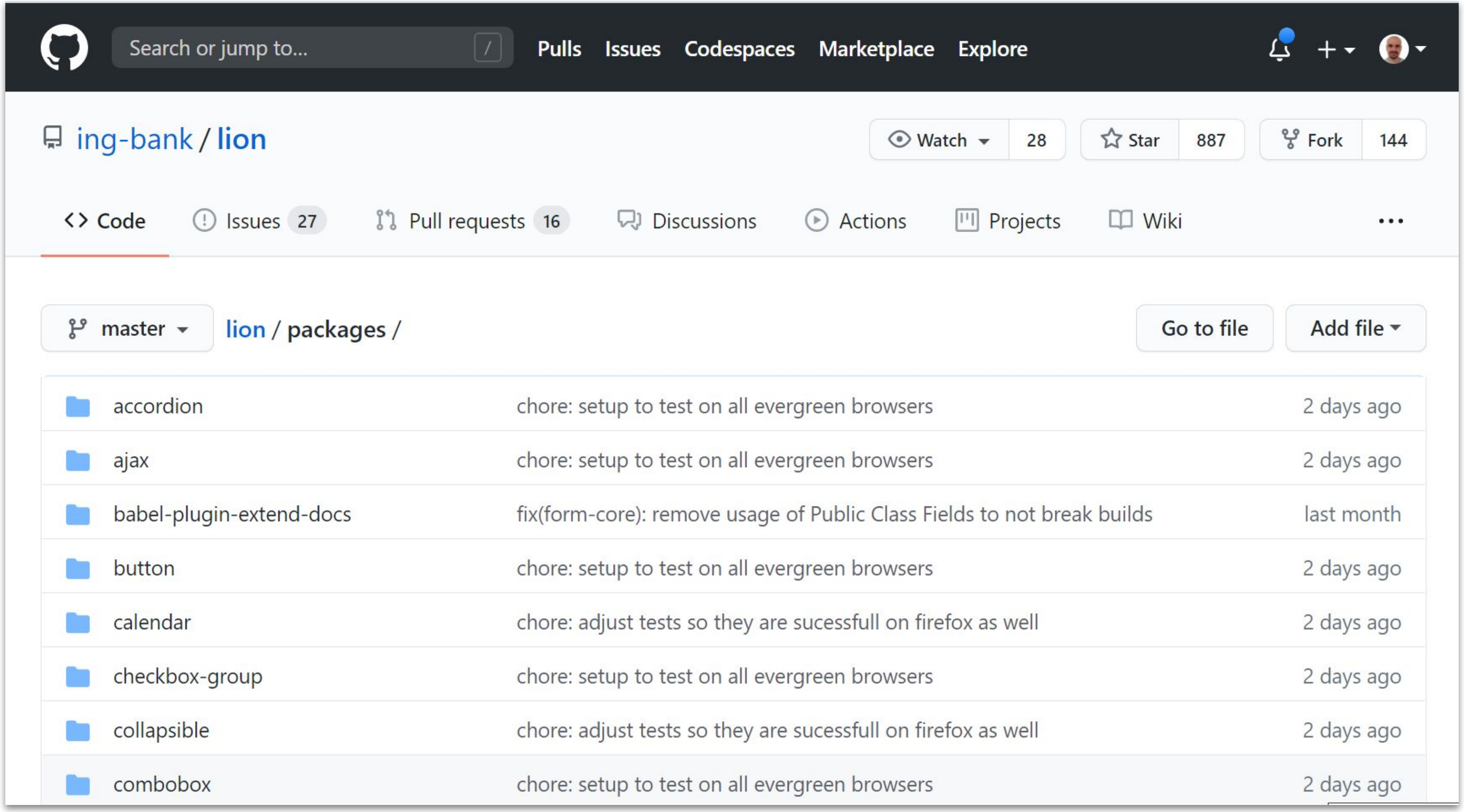


Theming and customizing components

How to organize the catalog

Packages, imports and pragmatism

A single repository



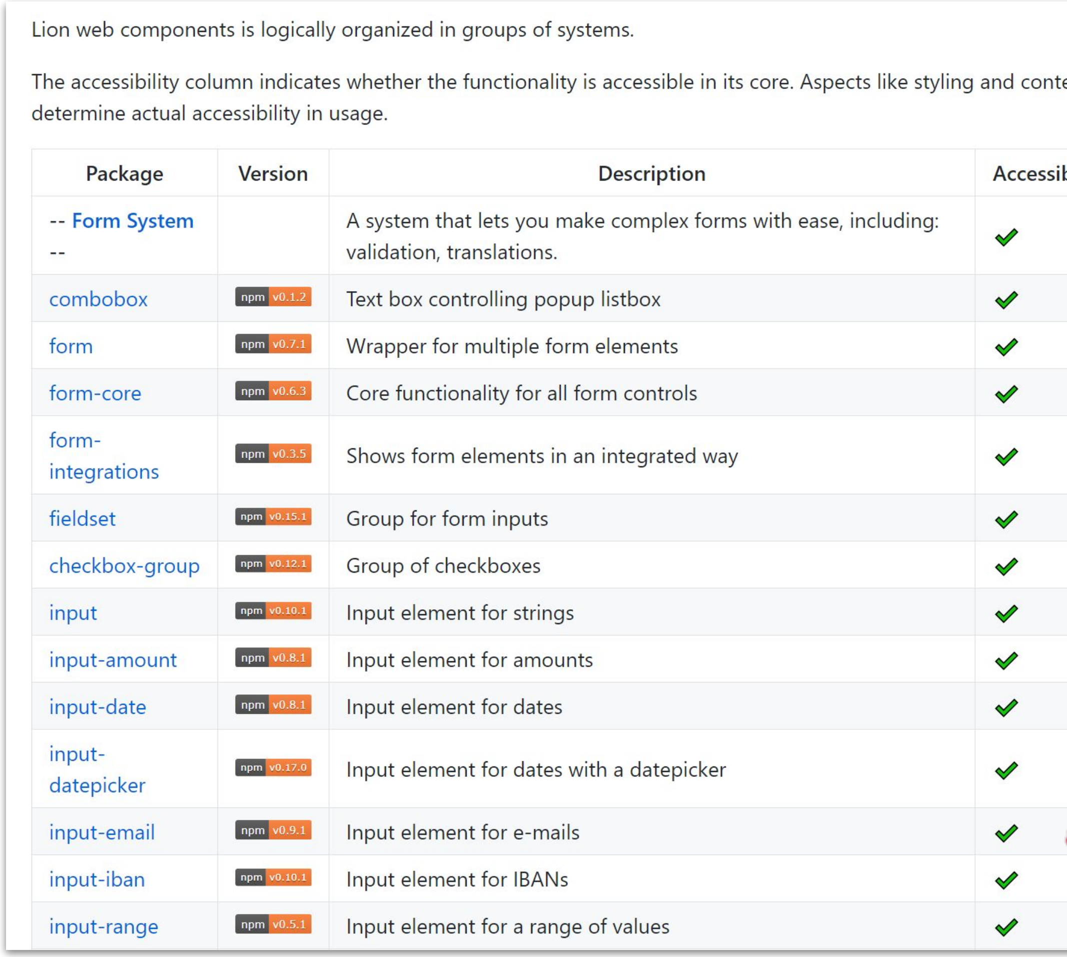
ing-bank / lion

lion / packages /

accordion	chore: setup to test on all evergreen browsers	2 days ago
ajax	chore: setup to test on all evergreen browsers	2 days ago
babel-plugin-extend-docs	fix(form-core): remove usage of Public Class Fields to not break builds	last month
button	chore: setup to test on all evergreen browsers	2 days ago
calendar	chore: adjust tests so they are sucessfull on firefox as well	2 days ago
checkbox-group	chore: setup to test on all evergreen browsers	2 days ago
collapsible	chore: adjust tests so they are sucessfull on firefox as well	2 days ago
combobox	chore: setup to test on all evergreen browsers	2 days ago

Single source of truth for the catalog

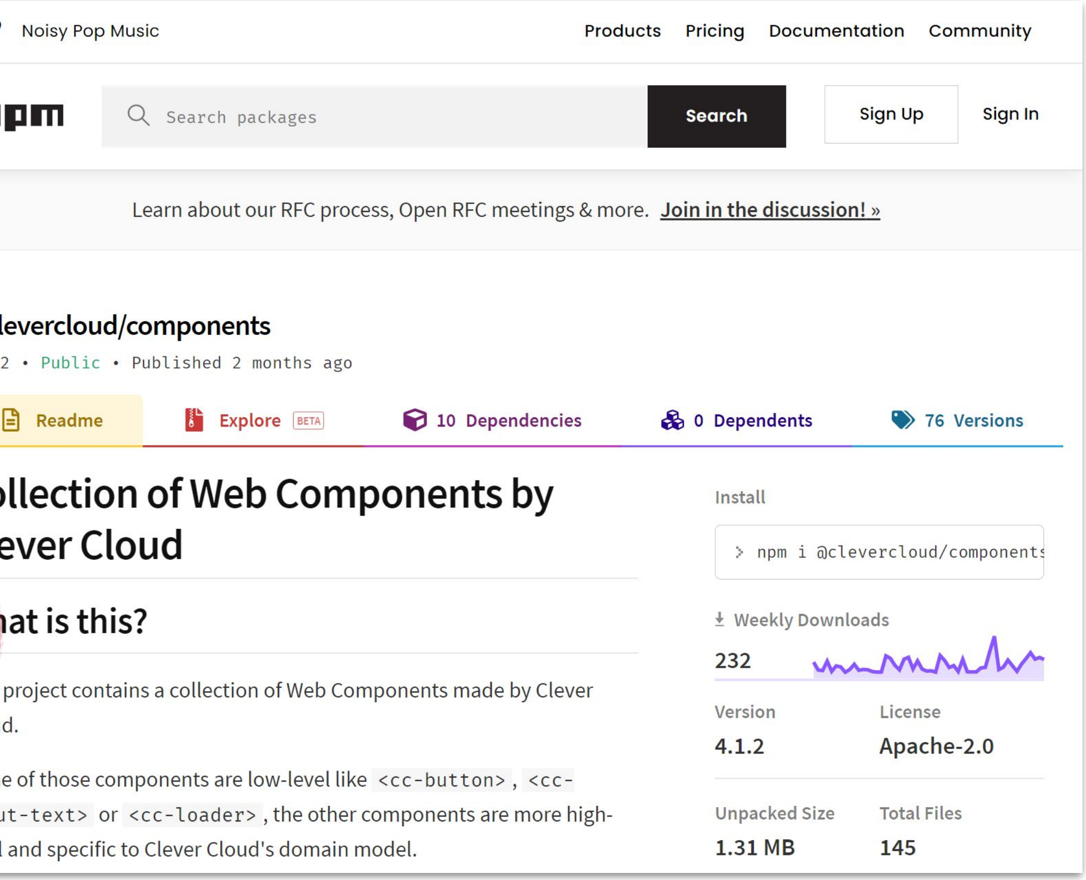
Two schools of thought



Lion web components is logically organized in groups of systems.

The accessibility column indicates whether the functionality is accessible in its core. Aspects like styling and content determine actual accessibility in usage.

Package	Version	Description	Accessibility
-- Form System --		A system that lets you make complex forms with ease, including: validation, translations.	✓
combobox	npm v0.1.2	Text box controlling popup listbox	✓
form	npm v0.7.1	Wrapper for multiple form elements	✓
form-core	npm v0.6.3	Core functionality for all form controls	✓
form-integrations	npm v0.3.5	Shows form elements in an integrated way	✓
fieldset	npm v0.15.1	Group for form inputs	✓
checkbox-group	npm v0.12.1	Group of checkboxes	✓
input	npm v0.10.1	Input element for strings	✓
input-amount	npm v0.8.1	Input element for amounts	✓
input-date	npm v0.8.1	Input element for dates	✓
input-datepicker	npm v0.17.0	Input element for dates with a datepicker	✓
input-email	npm v0.9.1	Input element for e-mails	✓
input-iban	npm v0.10.1	Input element for IBANs	✓
input-range	npm v0.5.1	Input element for a range of values	✓



Learn about our RFC process, Open RFC meetings & more. Join in the discussion! »

@clevercloud/components

4.1.2 • Public • Published 2 months ago

Collection of Web Components by Clever Cloud

What is this?

This project contains a collection of Web Components made by Clever Cloud.

Some of those components are low-level like `<cc-button>`, `<cc-input-text>` or `<cc-loader>`, the other components are more high-level and specific to Clever Cloud's domain model.

Install

```
> npm i @clevercloud/components
```

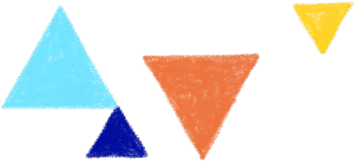
Weekly Downloads

232

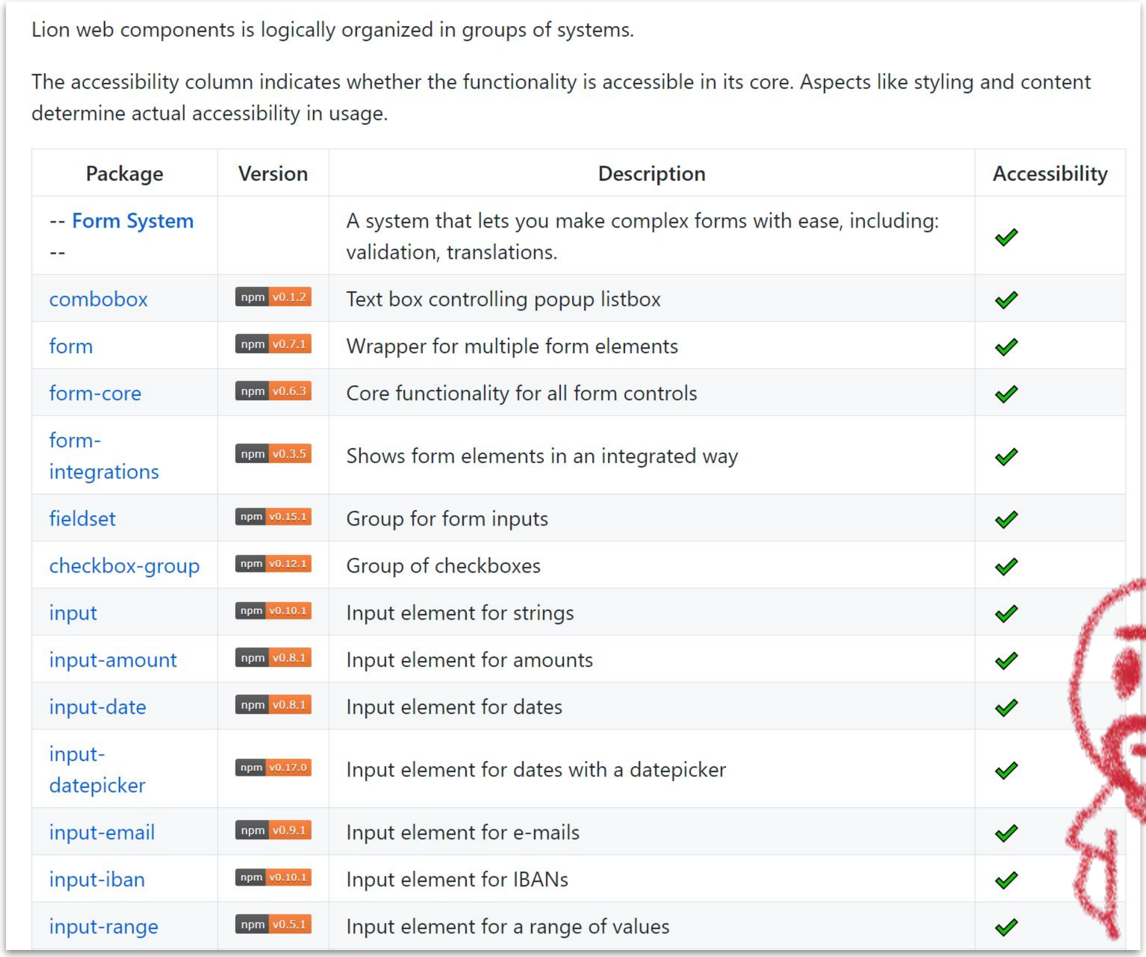
Version	License
4.1.2	Apache-2.0

Unpacked Size	Total Files
1.31 MB	145

A packet per component or a global one



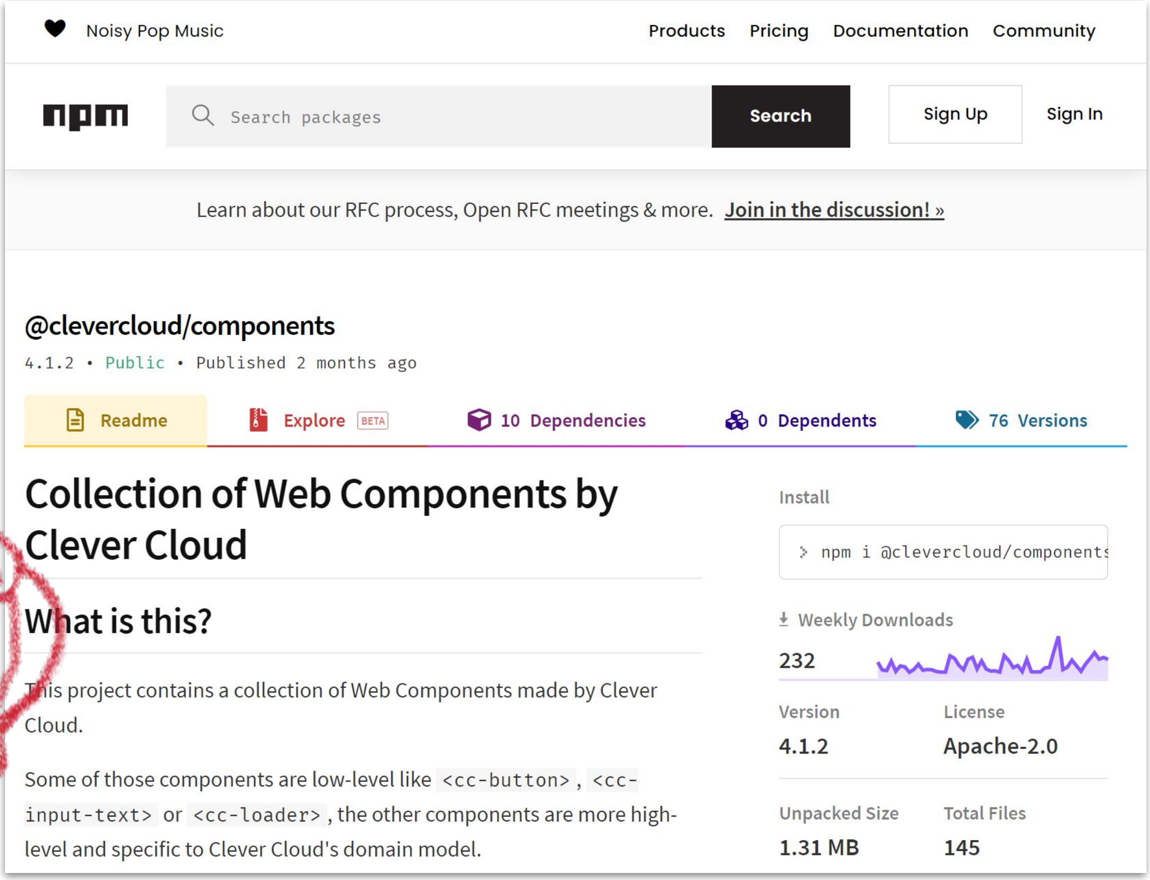
Two schools of thought



Lion web components is logically organized in groups of systems.

The accessibility column indicates whether the functionality is accessible in its core. Aspects like styling and content determine actual accessibility in usage.

Package	Version	Description	Accessibility
-- Form System --		A system that lets you make complex forms with ease, including: validation, translations.	✓
combobox	npm v0.1.2	Text box controlling popup listbox	✓
form	npm v0.7.1	Wrapper for multiple form elements	✓
form-core	npm v0.6.3	Core functionality for all form controls	✓
form-integrations	npm v0.3.5	Shows form elements in an integrated way	✓
fieldset	npm v0.15.1	Group for form inputs	✓
checkbox-group	npm v0.12.1	Group of checkboxes	✓
input	npm v0.10.1	Input element for strings	✓
input-amount	npm v0.8.1	Input element for amounts	✓
input-date	npm v0.8.1	Input element for dates	✓
input-datepicker	npm v0.17.0	Input element for dates with a datepicker	✓
input-email	npm v0.9.1	Input element for e-mails	✓
input-iban	npm v0.10.1	Input element for IBANs	✓
input-range	npm v0.5.1	Input element for a range of values	✓



Learn about our RFC process, Open RFC meetings & more. Join in the discussion! »

@clevercloud/components

4.1.2 • Public • Published 2 months ago

Collection of Web Components by Clever Cloud

What is this?

This project contains a collection of Web Components made by Clever Cloud.

Some of those components are low-level like `<cc-button>`, `<cc-input-text>` or `<cc-loader>`, the other components are more high-level and specific to Clever Cloud's domain model.

Install

```
> npm i @clevercloud/components
```

Weekly Downloads

232

Version	License
4.1.2	Apache-2.0

Unpacked Size	Total Files
1.31 MB	145

Individual versioning vs global one

Lots of web components libraries

slim.js

LitElement

hybrids

snuggsi ツ

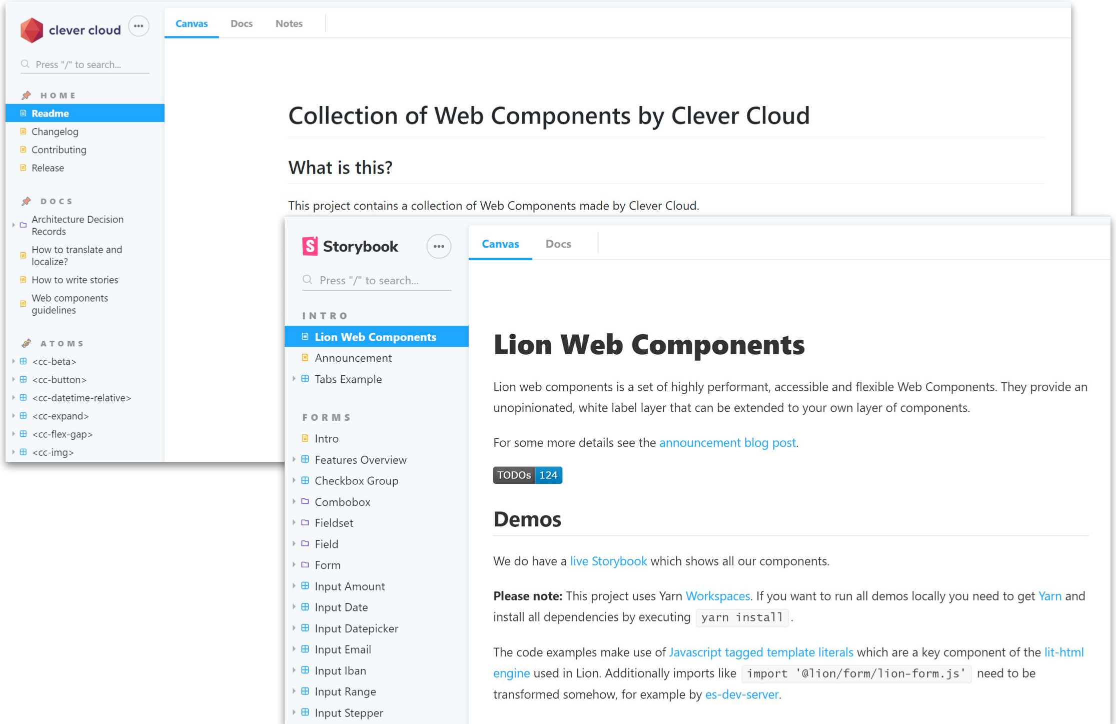
SkateJS

stencil

smart
HTML ELEMENTS

For different needs and sensibilities

Which ones to use?



Collection of Web Components by Clever Cloud

What is this?

This project contains a collection of Web Components made by Clever Cloud.

Lion Web Components

Lion web components is a set of highly performant, accessible and flexible Web Components. They provide an unopinionated, white label layer that can be extended to your own layer of components.

For some more details see the announcement blog post.

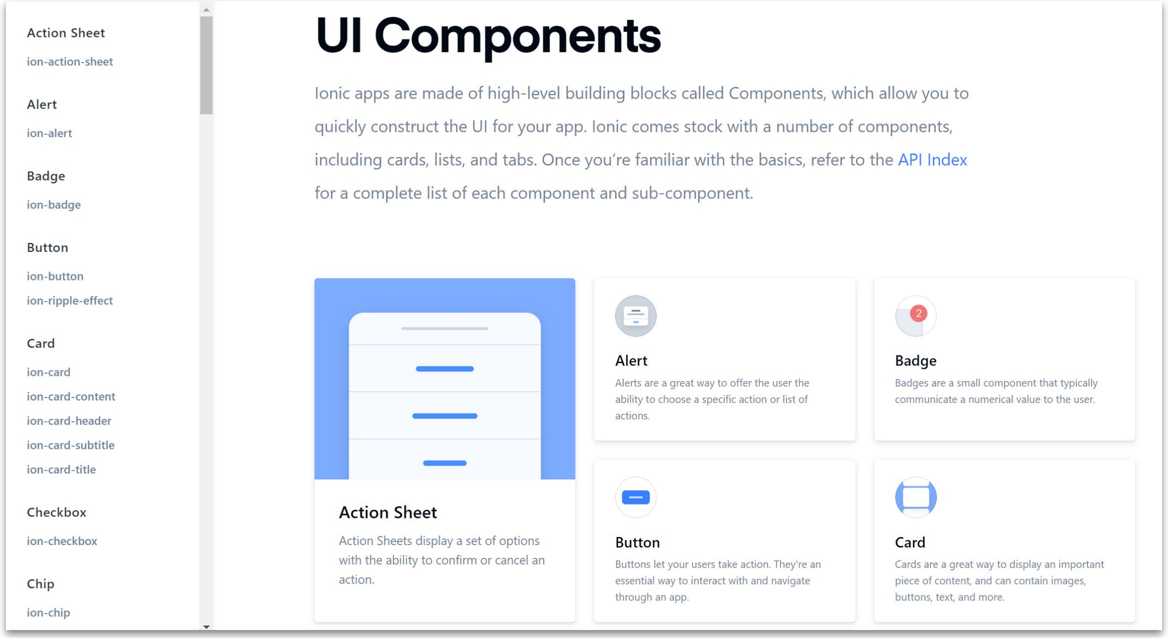
TODOs 124

Demos

We do have a live Storybook which shows all our components.

Please note: This project uses Yarn Workspaces. If you want to run all demos locally you need to get Yarn and install all dependencies by executing `yarn install`.

The code examples make use of Javascript tagged template literals which are a key component of the lit-html engine used in Lion. Additionally imports like `import '@lion/form/lion-form.js'` need to be transformed somehow, for example by es-dev-server.



UI Components

Ionic apps are made of high-level building blocks called Components, which allow you to quickly construct the UI for your app. Ionic comes stock with a number of components, including cards, lists, and tabs. Once you're familiar with the basics, refer to the API Index for a complete list of each component and sub-component.

Action Sheet
Action Sheets display a set of options with the ability to confirm or cancel an action.

Alert
Alerts are a great way to offer the user the ability to choose a specific action or list of actions.

Badge
Badges are a small component that typically communicate a numerical value to the user.

Button
Buttons let your users take action. They're an essential way to interact with and navigate through an app.

Card
Cards are a great way to display an important piece of content, and can contain images, buttons, text, and more.

LitElement

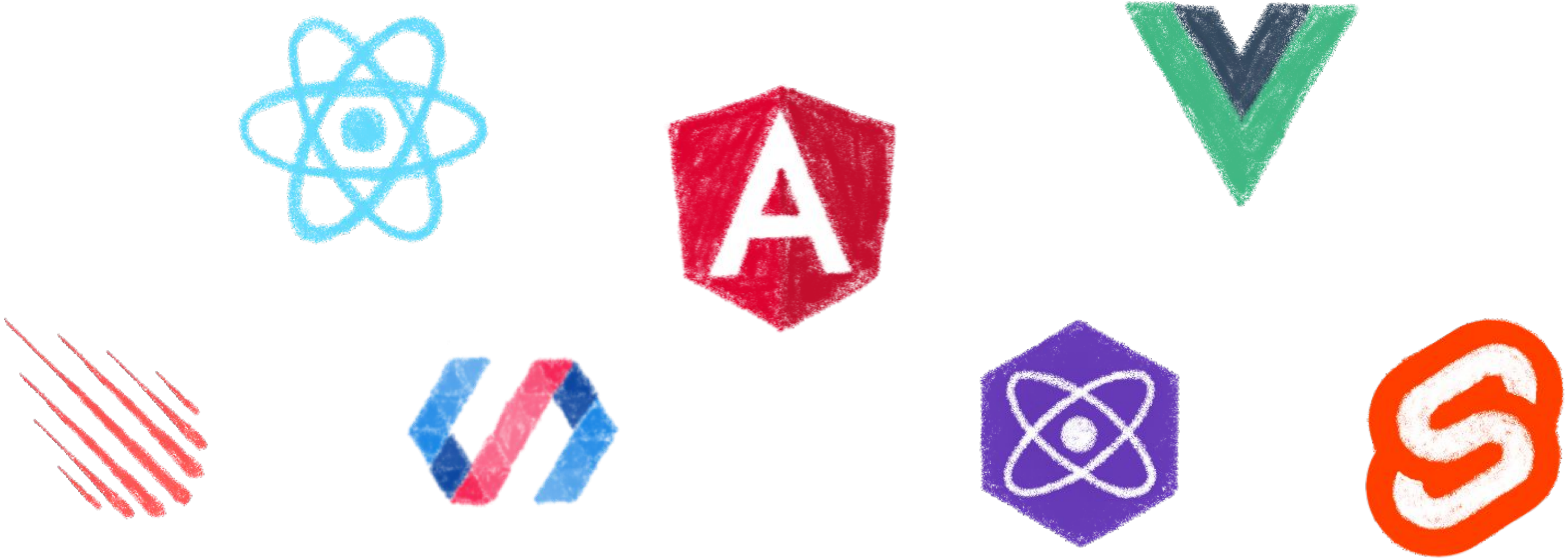
stencil

All are good, but these are popular favorites

Driving-up adoption

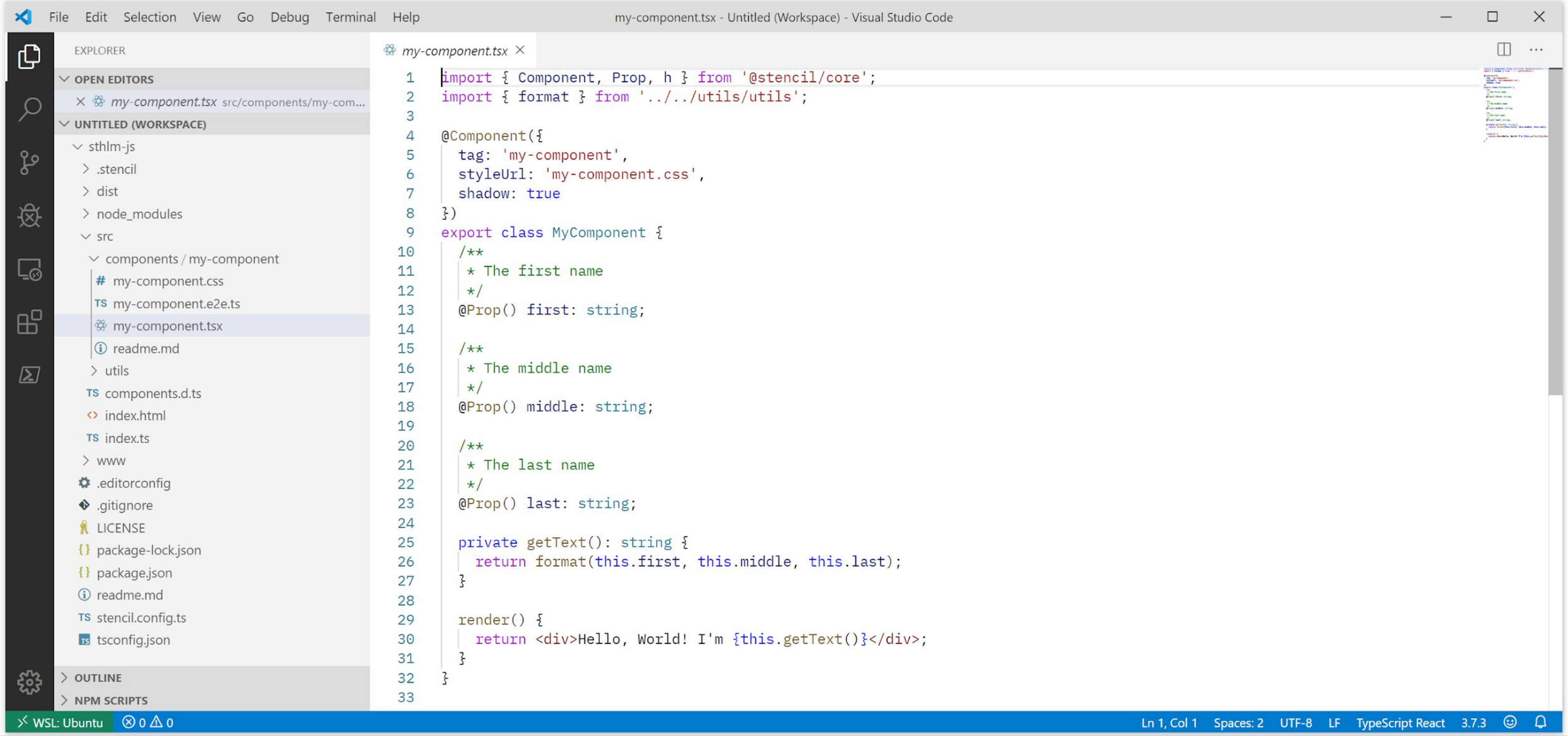
Making devs use your components

Think who are your target users



Users of any framework current or future...

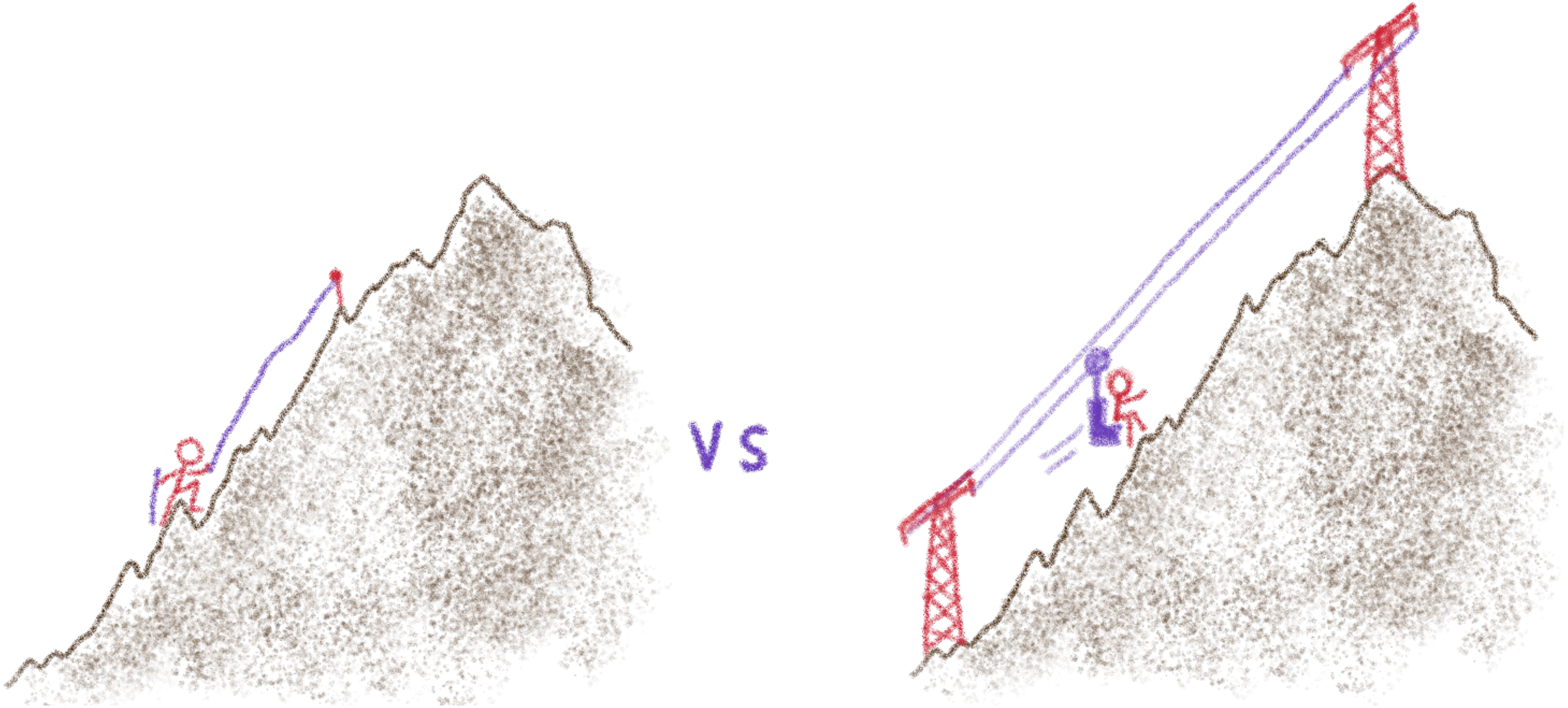
They aren't used to your library



```
import { Component, Prop, h } from '@stencil/core';
import { format } from '../../utils/utils';

@Component({
  tag: 'my-component',
  styleUrl: 'my-component.css',
  shadow: true
})
export class MyComponent {
  /**
   * The first name
   */
  @Prop() first: string;

  /**
   * The middle name
   */
  @Prop() middle: string;

  /**
   * The last name
   */
  @Prop() last: string;

  private getText(): string {
    return format(this.first, this.middle, this.last);
  }

  render() {
    return <div>Hello, World! I'm {this.getText()}</div>;
  }
}
```

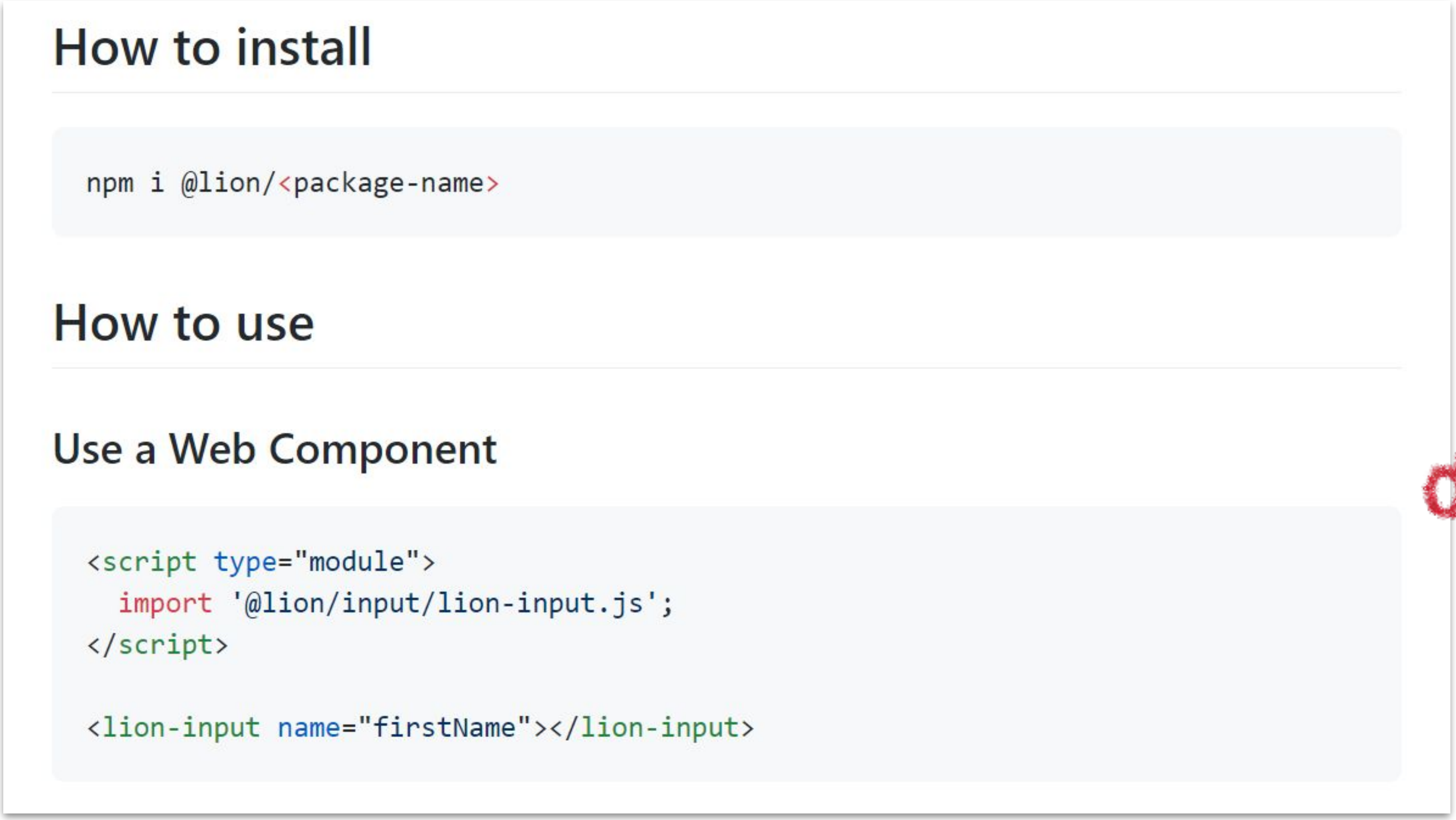
And they shouldn't need to be

Go the extra mile to drive up adoption



So they don't need to do it

Make it easy to use



How to install

```
npm i @lion/<package-name>
```

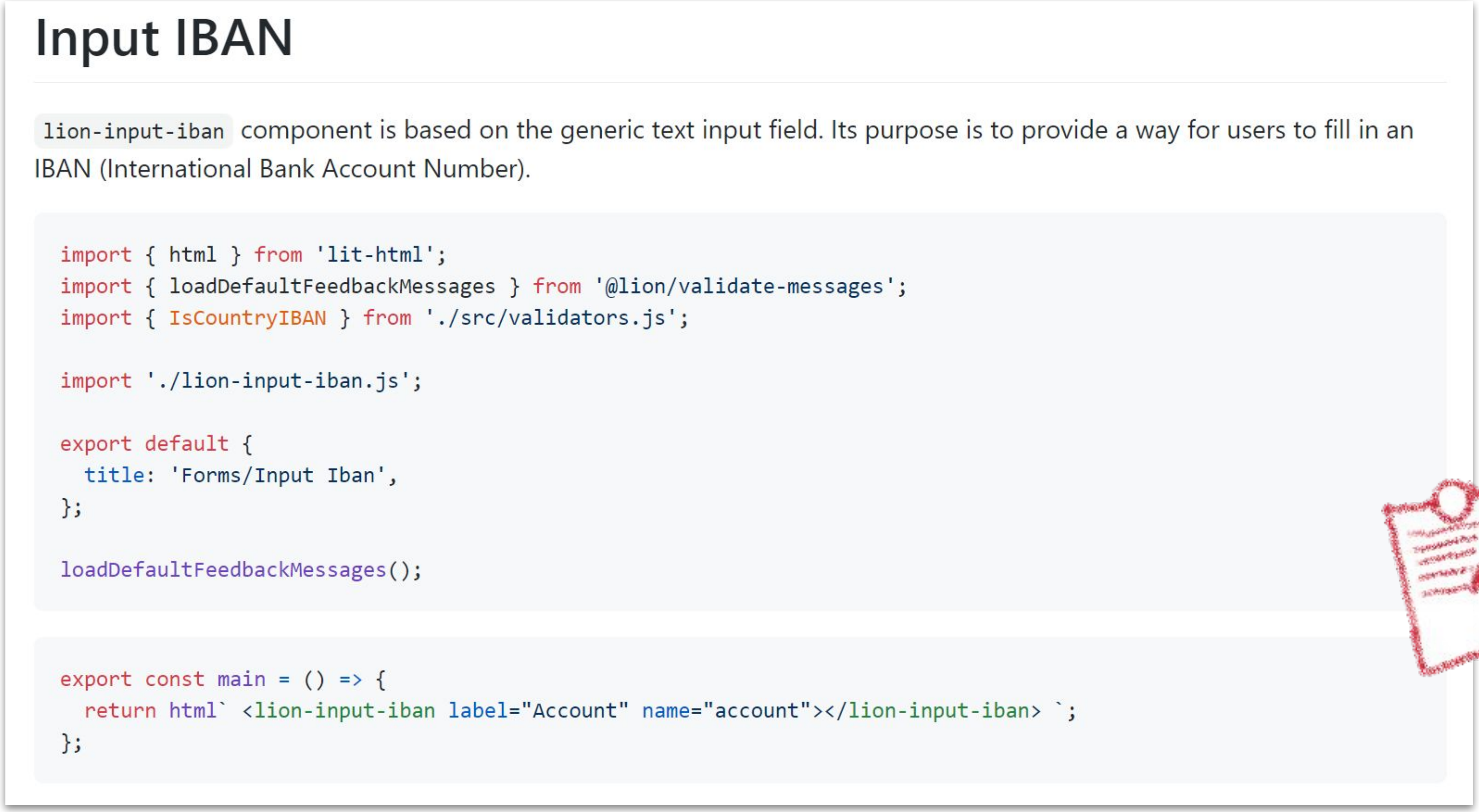
How to use

Use a Web Component

```
<script type="module">
  import '@lion/input/lion-input.js';
</script>

<lion-input name="firstName"></lion-input>
```

As easy as a HTML tag

Document every composant



Input IBAN

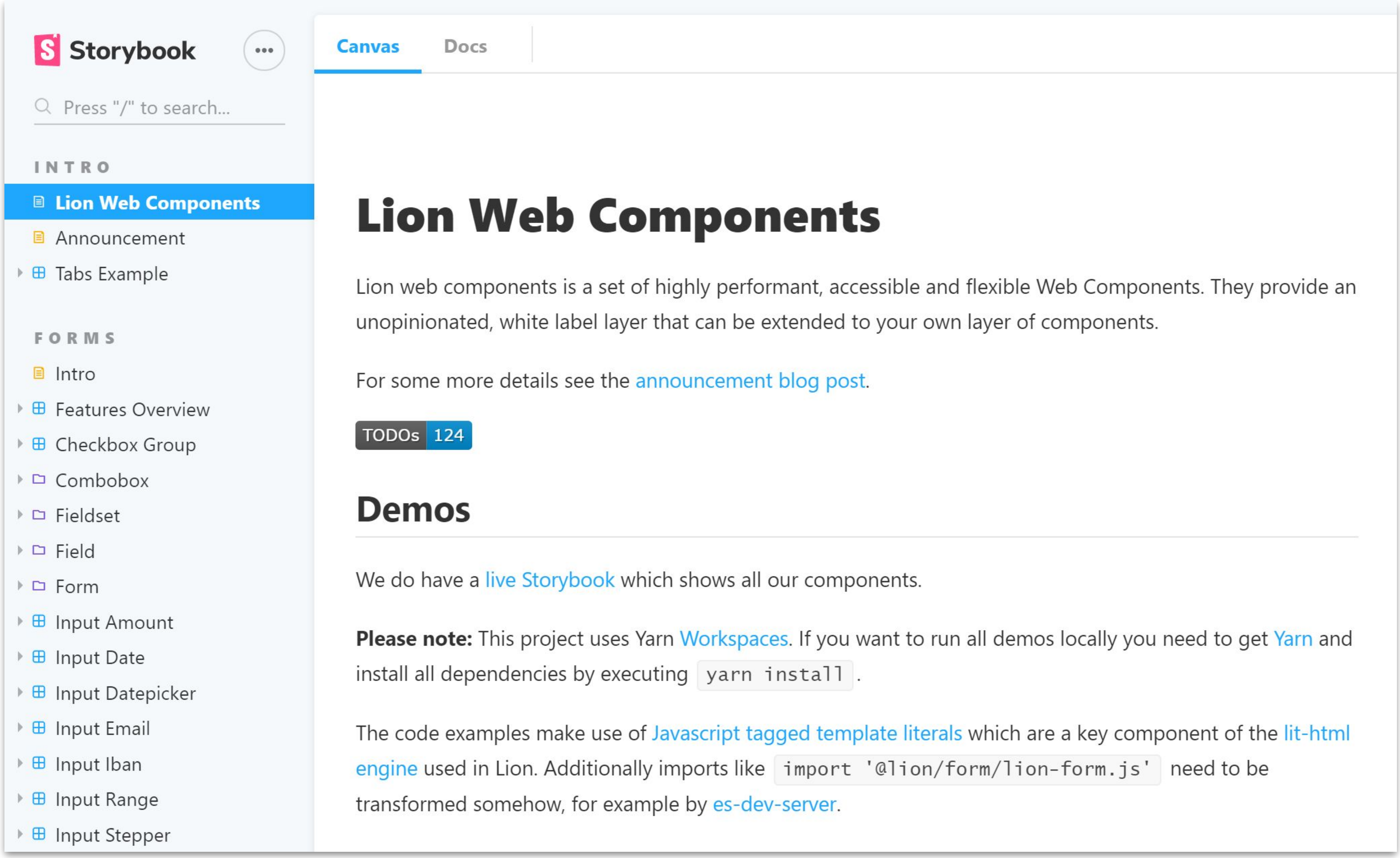
`lion-input-iban` component is based on the generic text input field. Its purpose is to provide a way for users to fill in an IBAN (International Bank Account Number).

```
import { html } from 'lit-html';
import { loadDefaultFeedbackMessages } from '@lion/validate-messages';
import { IsCountryIBAN } from './src/validators.js';

import './lion-input-iban.js';

export default {
  title: 'Forms/Input Iban',
};

loadDefaultFeedbackMessages();
```

```
export const main = () => {
  return html` <lion-input-iban label="Account" name="account"></lion-input-iban> `;
};
```

How to use, inputs/outputs, examples...

Documentation isn't enough



Lion Web Components

Lion web components is a set of highly performant, accessible and flexible Web Components. They provide an unopinionated, white label layer that can be extended to your own layer of components.

For some more details see the announcement blog post.

TODOs 124

Demos

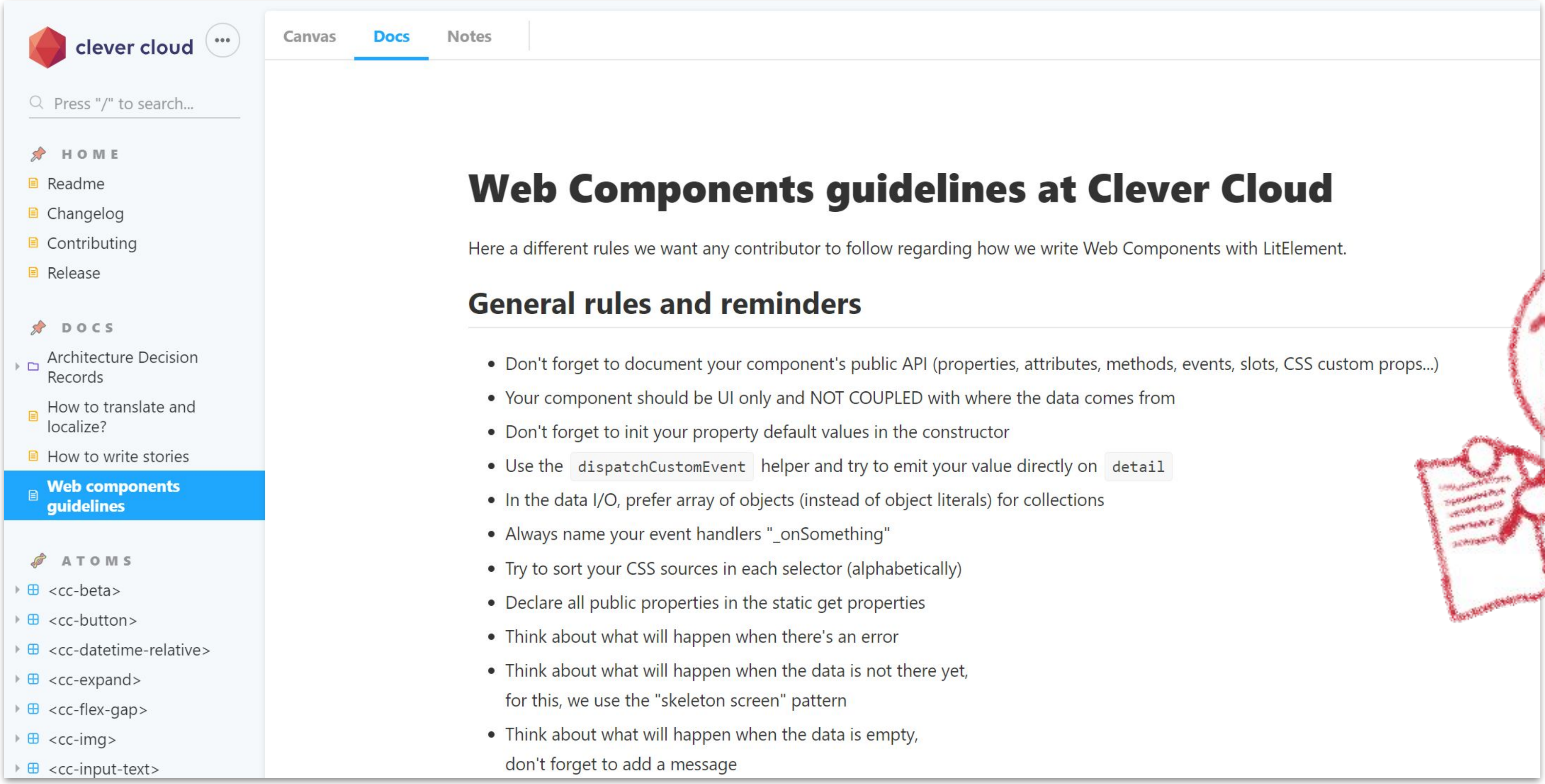
We do have a live Storybook which shows all our components.

Please note: This project uses Yarn Workspaces. If you want to run all demos locally you need to get Yarn and install all dependencies by executing `yarn install`.

The code examples make use of Javascript tagged template literals which are a key component of the lit-html engine used in Lion. Additionally imports like `import '@lion/form/lion-form.js'` need to be transformed somehow, for example by es-dev-server.

Storybook make adoption easy

Keeping a coherent writing style

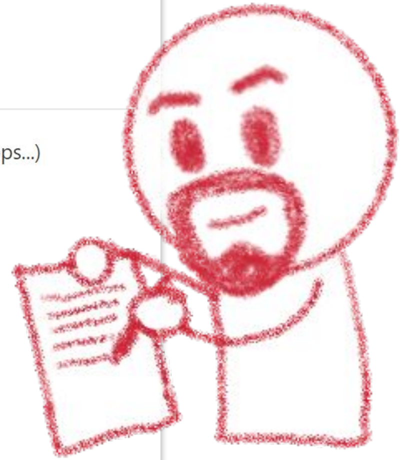


Web Components guidelines at Clever Cloud

Here a different rules we want any contributor to follow regarding how we write Web Components with LitElement.

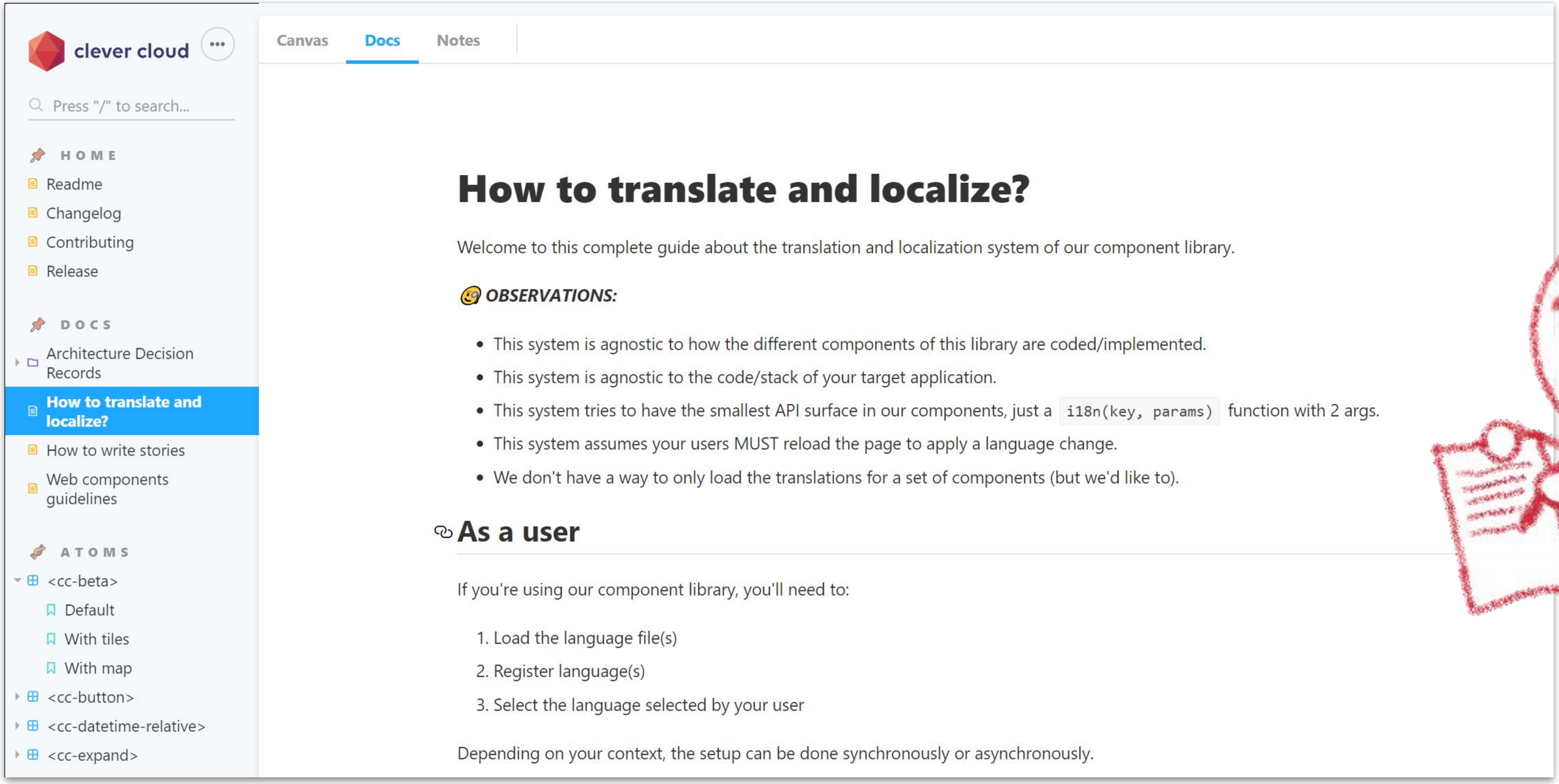
General rules and reminders

- Don't forget to document your component's public API (properties, attributes, methods, events, slots, CSS custom props...)
- Your component should be UI only and NOT COUPLED with where the data comes from
- Don't forget to init your property default values in the constructor
- Use the `dispatchCustomEvent` helper and try to emit your value directly on `detail`
- In the data I/O, prefer array of objects (instead of object literals) for collections
- Always name your event handlers "_onSomething"
- Try to sort your CSS sources in each selector (alphabetically)
- Declare all public properties in the static get properties
- Think about what will happen when there's an error
- Think about what will happen when the data is not there yet, for this, we use the "skeleton screen" pattern
- Think about what will happen when the data is empty, don't forget to add a message



Write down your guidelines

I18n shouldn't be an afterthought



How to translate and localize?

Welcome to this complete guide about the translation and localization system of our component library.

🧐 ***OBSERVATIONS:***

- This system is agnostic to how the different components of this library are coded/implemented.
- This system is agnostic to the code/stack of your target application.
- This system tries to have the smallest API surface in our components, just a `i18n(key, params)` function with 2 args.
- This system assumes your users MUST reload the page to apply a language change.
- We don't have a way to only load the translations for a set of components (but we'd like to).

As a user

If you're using our component library, you'll need to:

1. Load the language file(s)
2. Register language(s)
3. Select the language selected by your user

Depending on your context, the setup can be done synchronously or asynchronously.

Prepare everything for internationalization

That's all, folks!

Thank you all!