

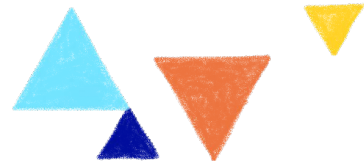
Design, develop and manage a catalog of Web Components

Horacio Gonzalez - @LostInBrittany

{CODE}motion
Online Tech Conference #!

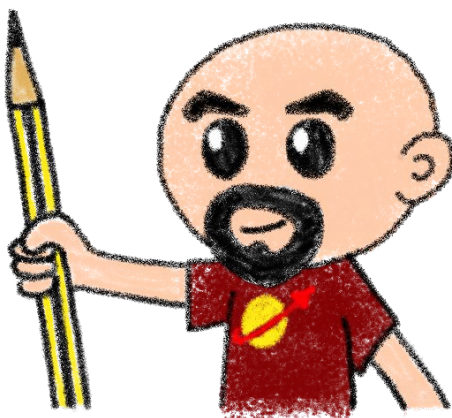
March, 23-24-25, 2021

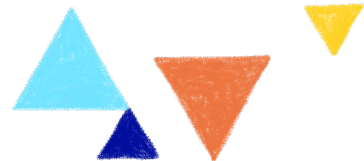




Who are we?

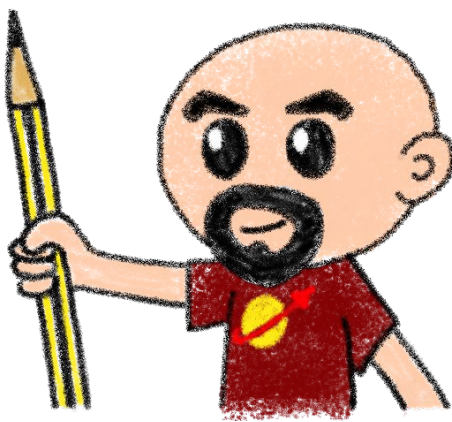
Introducing myself and
introducing ~~OVH~~ OVHcloud





Who are we?

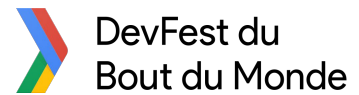
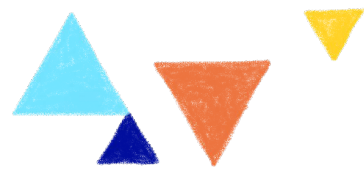
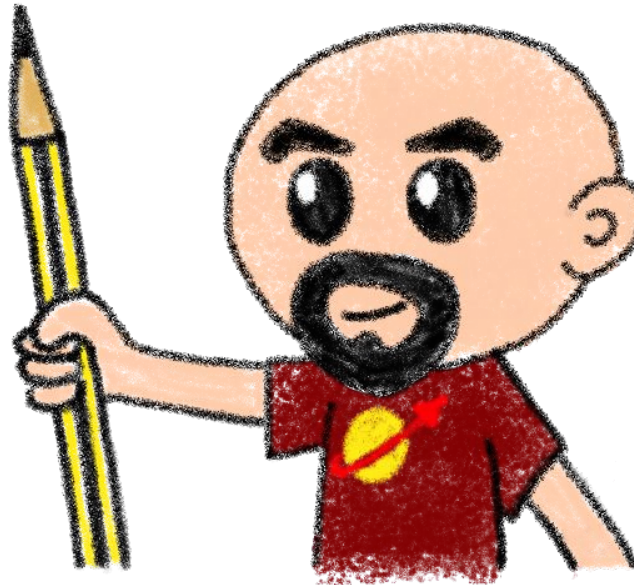
Introducing myself and
introducing ~~OVH~~ OVHcloud



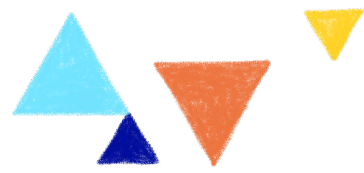
Horacio Gonzalez

@LostInBrittany

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



OVHcloud: A global leader



Web Cloud & Telecom



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

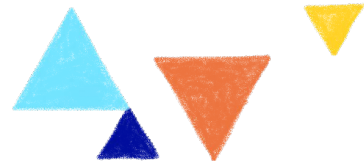


P.U.E. 1.09
Energy efficiency indicator



20 Years in Business
Disrupting since 1999



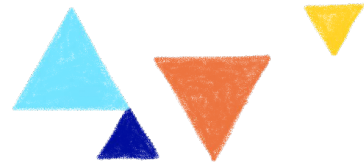


The 3 minutes context

What the heck are web component?



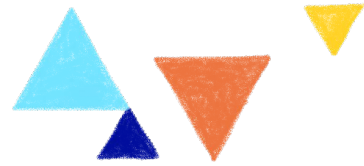
Web Components



Web standard W3C



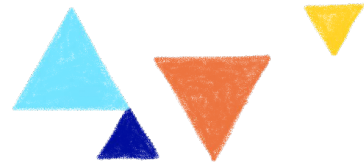
Web Components



Available in all modern browsers:
Firefox, Safari, Chrome



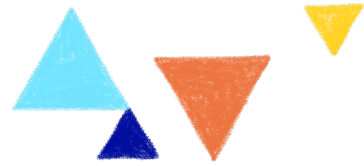
Web Components



Create your own HTML tags
Encapsulating look and behavior



Web Components

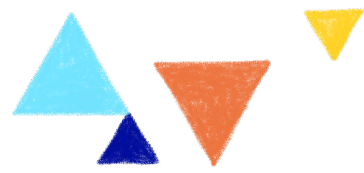


Fully interoperable

With other web components, with any framework



Web Components



CUSTOM ELEMENTS



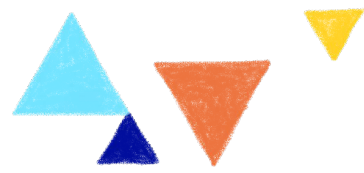
SHADOW DOM



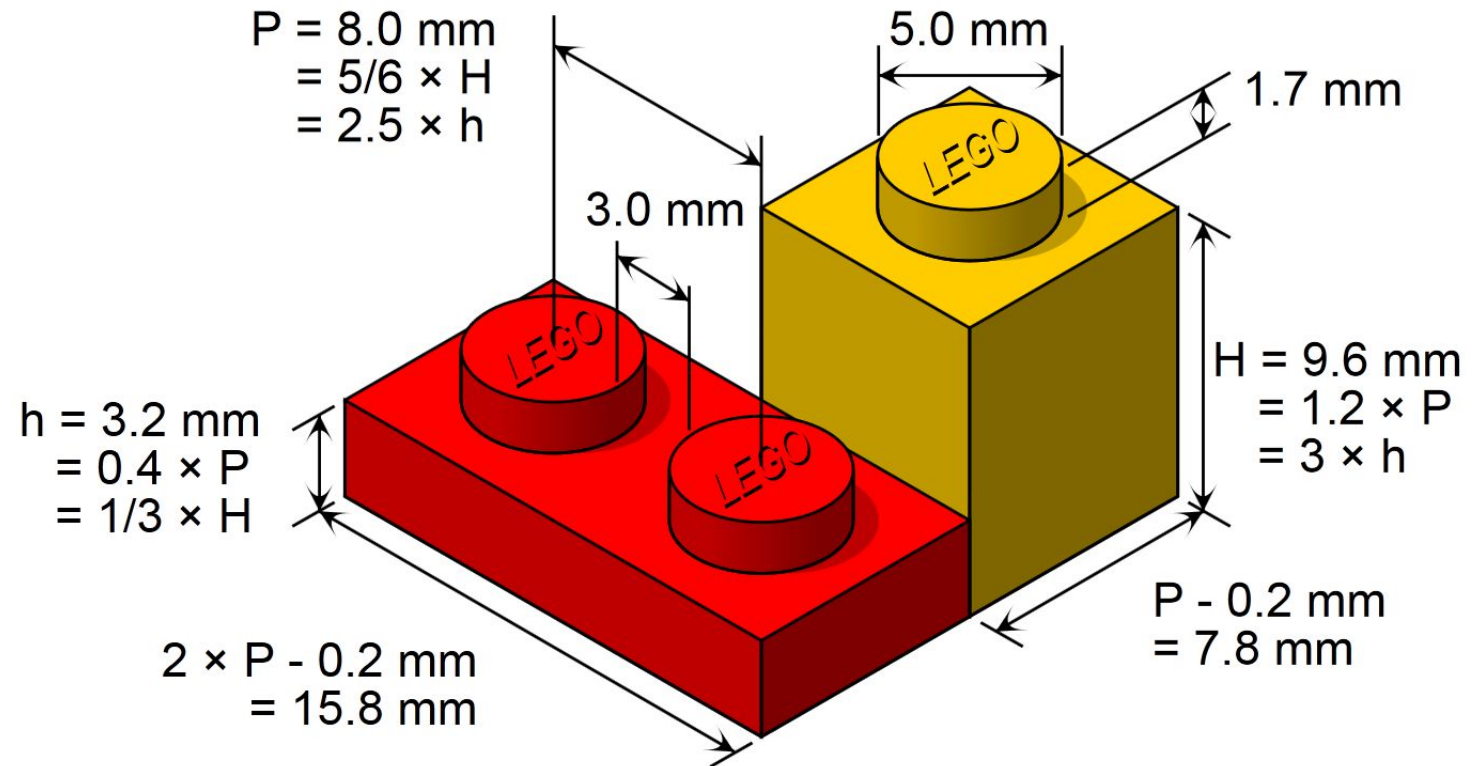
TEMPLATES

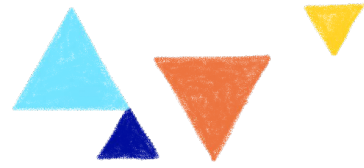


But in fact, it's just an element...



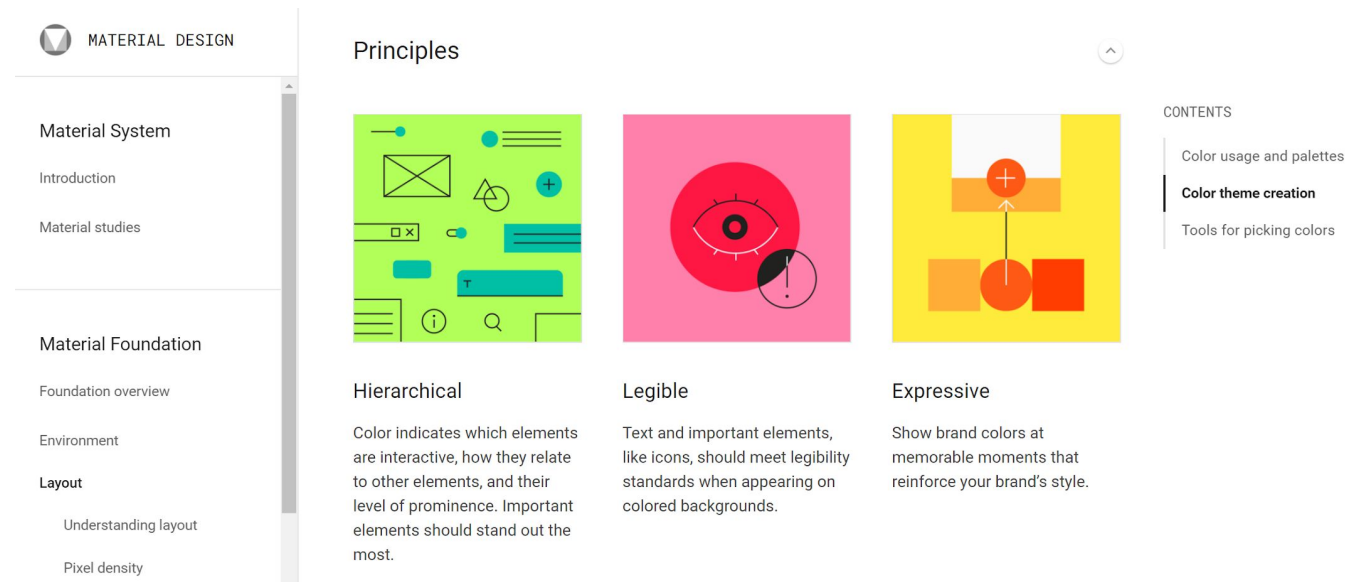
- Attributes
- Properties
- Methods
- Events





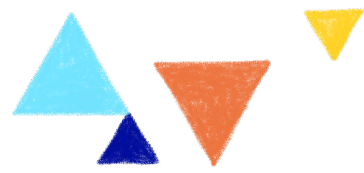
So, what are Design Systems?

And why should I look at them?





The same or different?



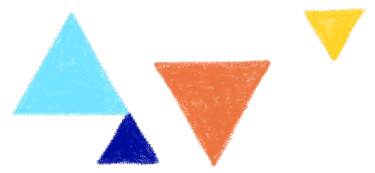
Design System

Component Catalog

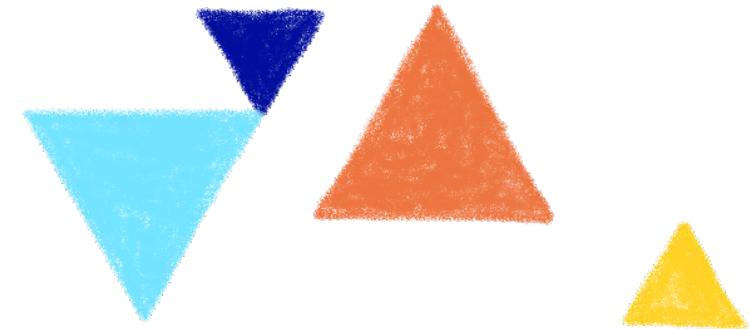
Style Guide



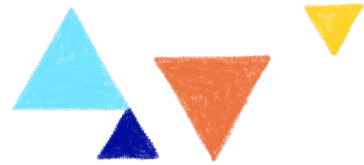
Style Guides



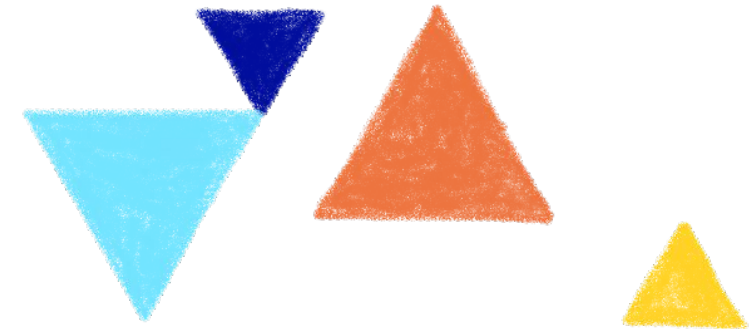
A **document** listing the **styles**, **patterns**, **practices**,
and **principles** of a brand **design standards**



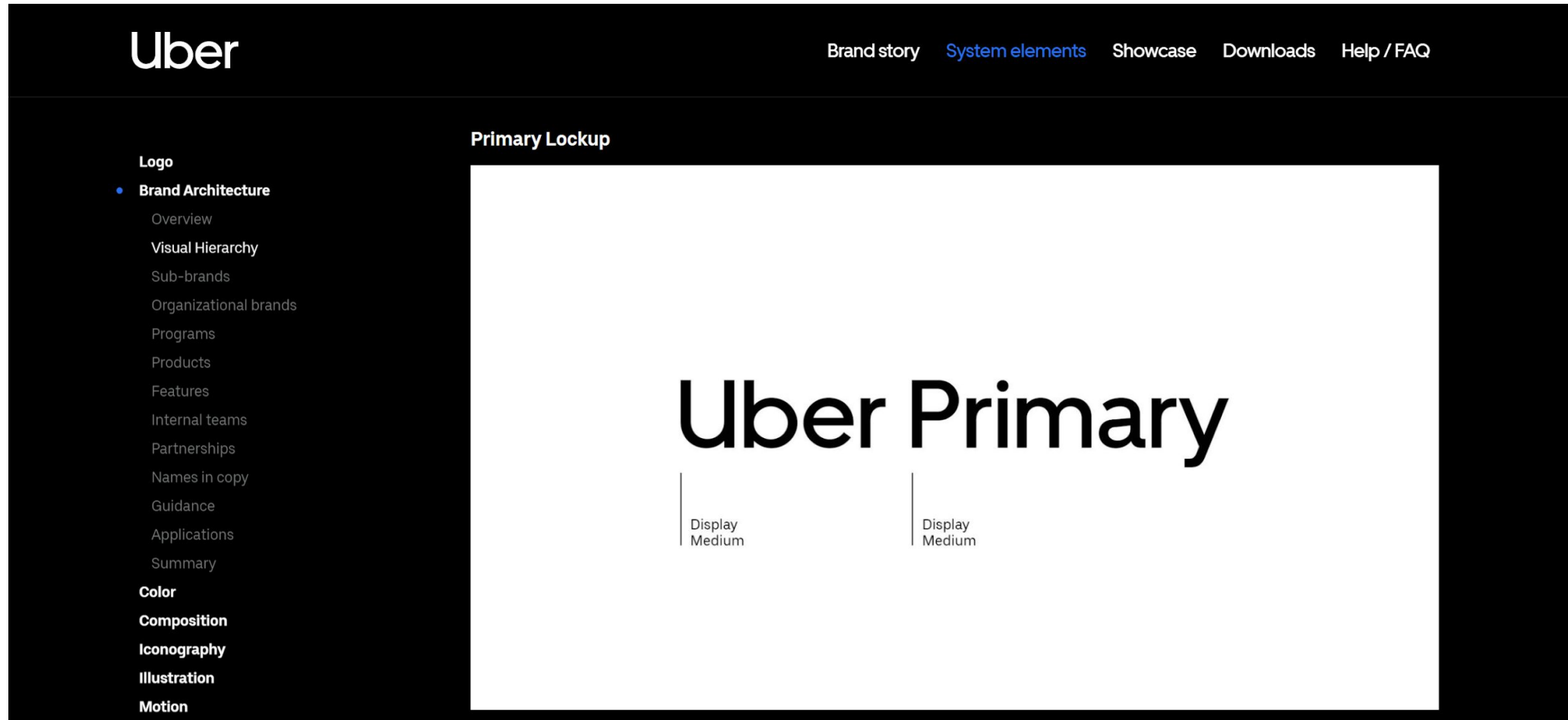
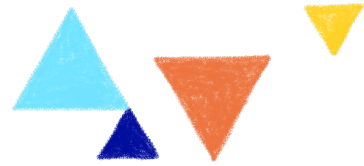
Style Guides



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

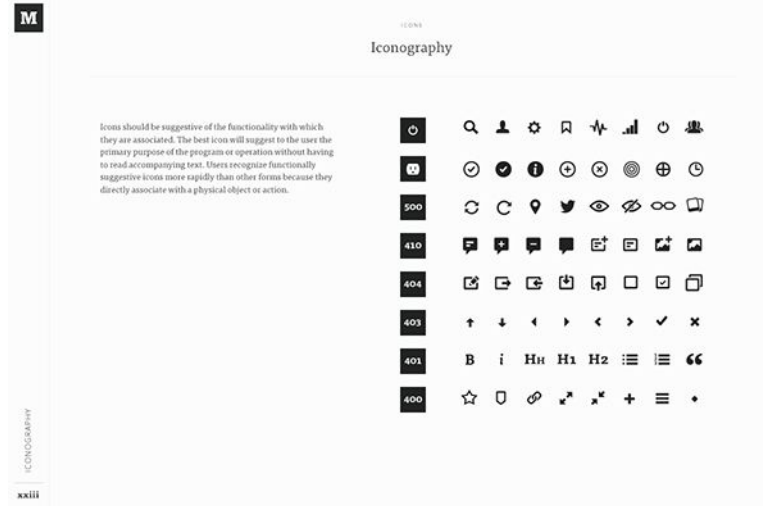
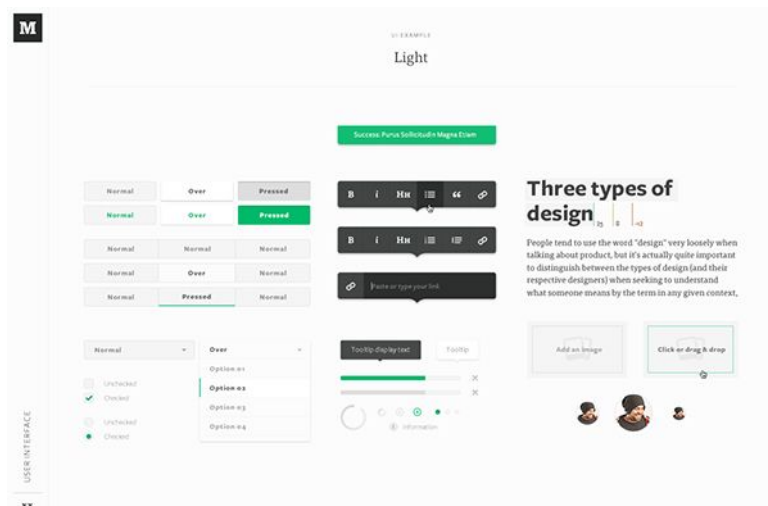
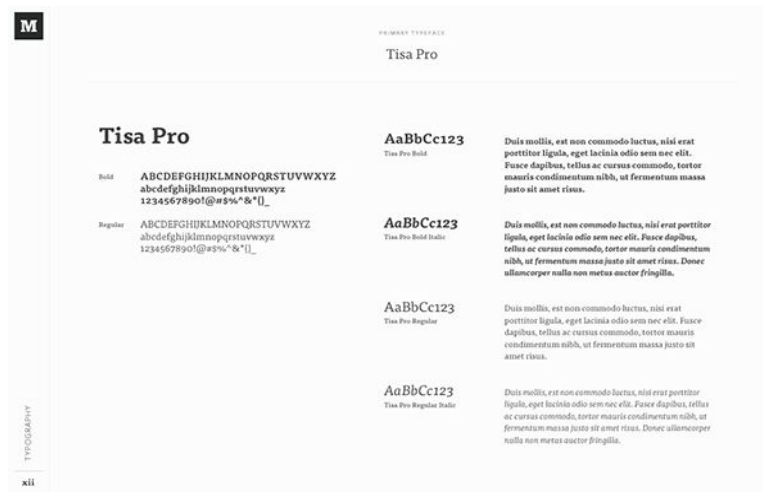


Style Guide Example: Uber

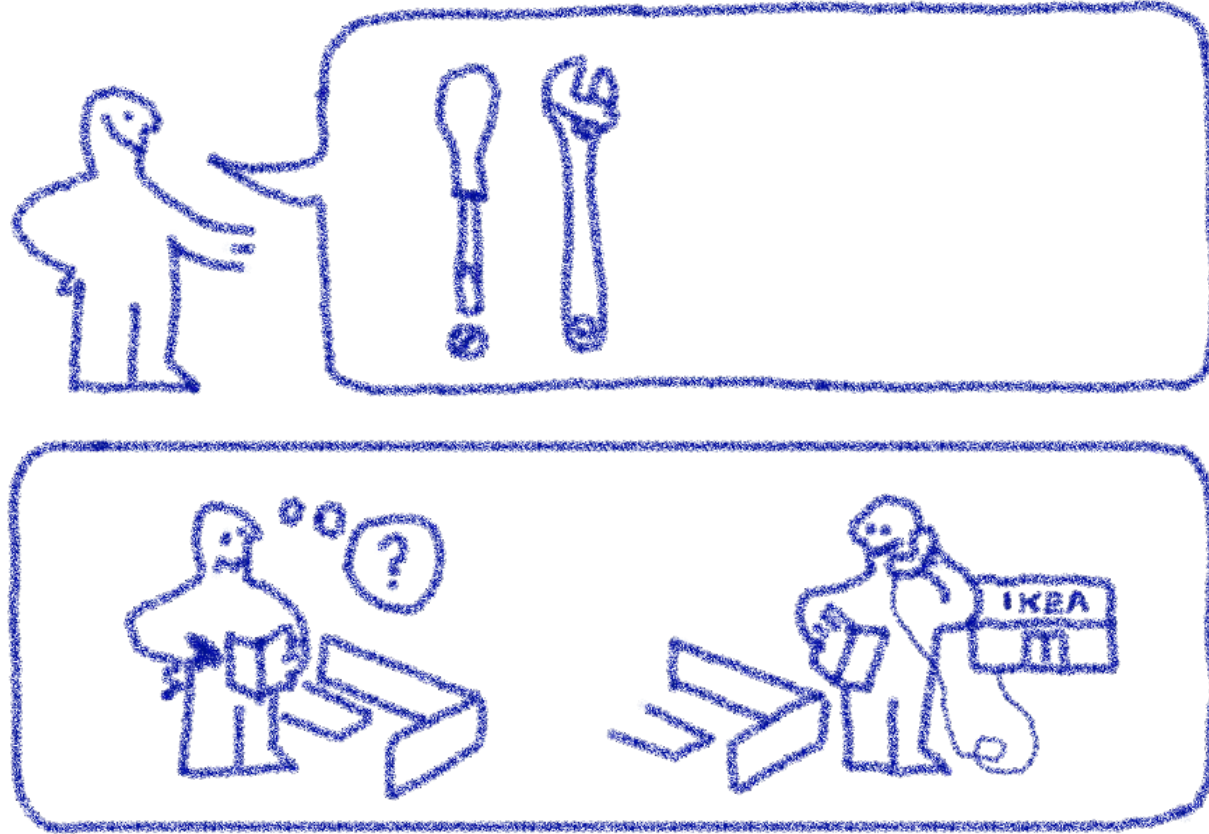
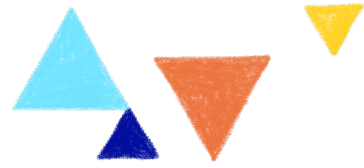


<https://brand.uber.com/>





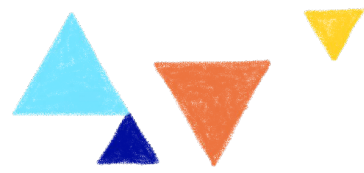
Style Guides alone are ambiguous



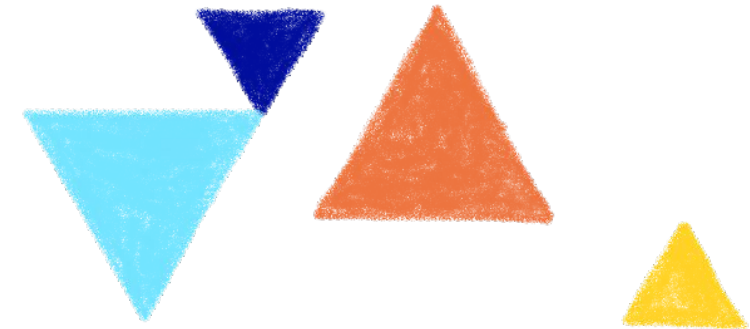
Interpretation needed to adapt
the preconisation to the use case



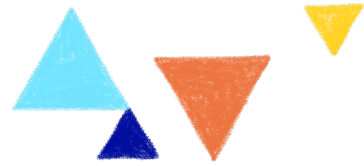
Component Catalogs



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!

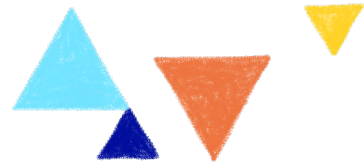
```
<div class="alert alert-primary" role="alert">
  A simple primary alert—check it out!
</div>
<div class="alert alert-secondary" role="alert">
  A simple secondary alert—check it out!
</div>
<div class="alert alert-success" role="alert">
  A simple success alert—check it out!
</div>
```



<https://getbootstrap.com/>



Component Catalog Example: ING's Lion



The screenshot shows the Storybook interface for the 'Date' component. The left sidebar lists the component catalog structure:

- INTRO
 - Lion Web Components
 - Announcement
 - Tabs Example
- FORMS
 - Intro
 - Features Overview
 - Checkbox Group
 - Fieldset
 - Field
 - Form
 - Input Amount
 - Input Date
 - Input Datepicker
 - Main (selected)
 - Minimum And Maximum Date
 - Disable Specific Dates
 - Calendar Heading
 - Disabled
 - Read Only

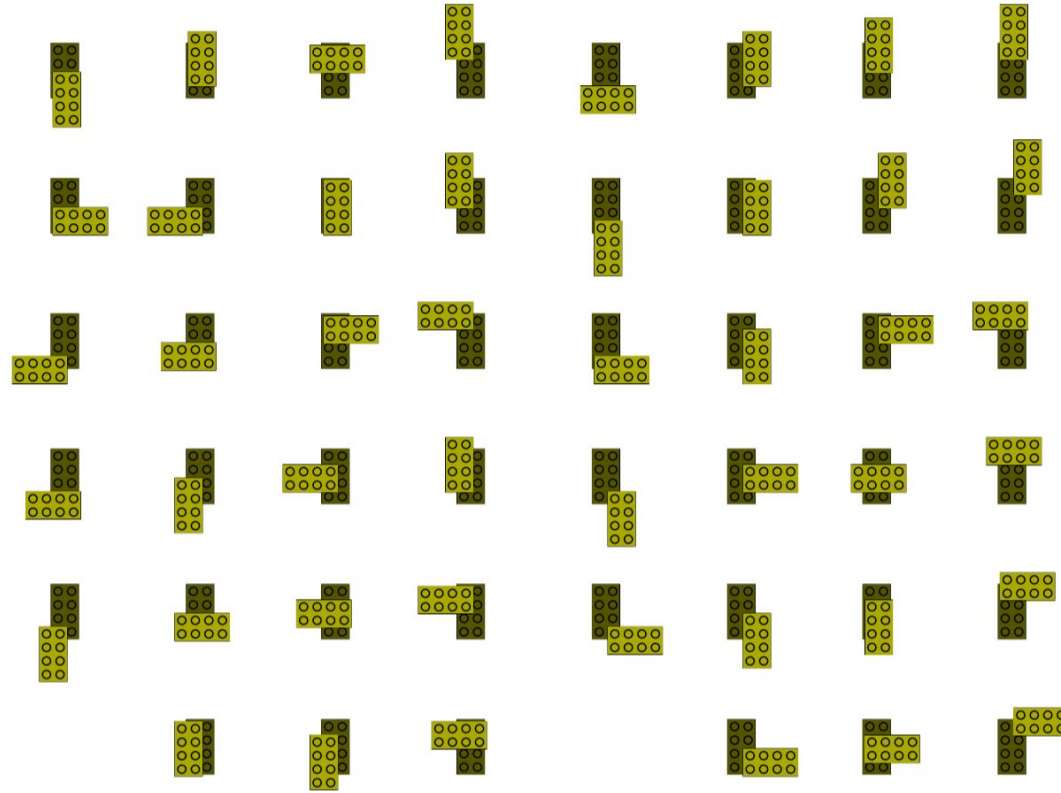
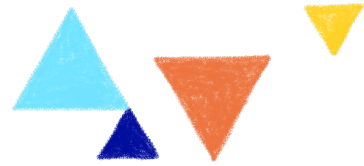
The main canvas displays the 'Date' component, which is a calendar for June 2020. The date '22' is highlighted with a green border.

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	<u>22</u>	23	24	25	26	27
28	29	30	1	2	3	4

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



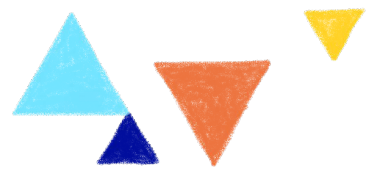
Catalogs alone create inconsistency



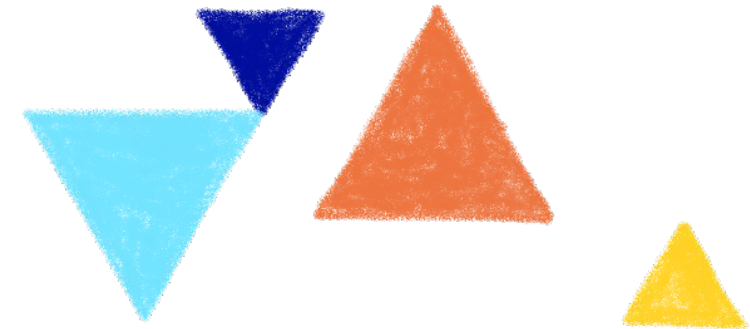
Like using the same LEGO bricks
to create very different objects



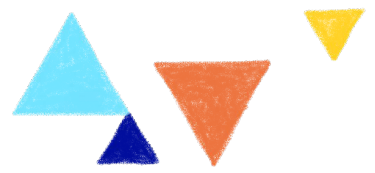
Design Systems



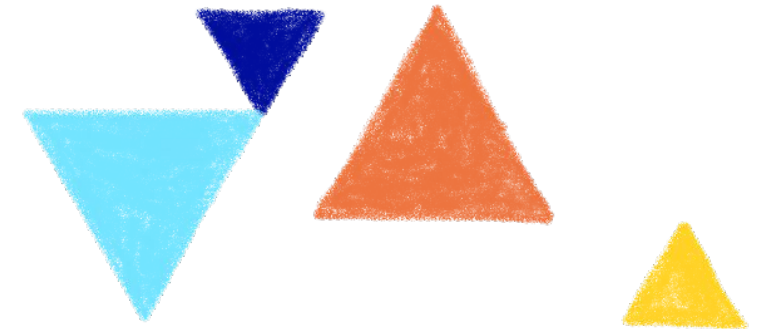
A Design System is like a **common visual language** for **product teams**



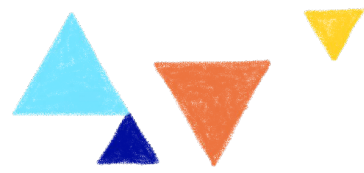
Design systems



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



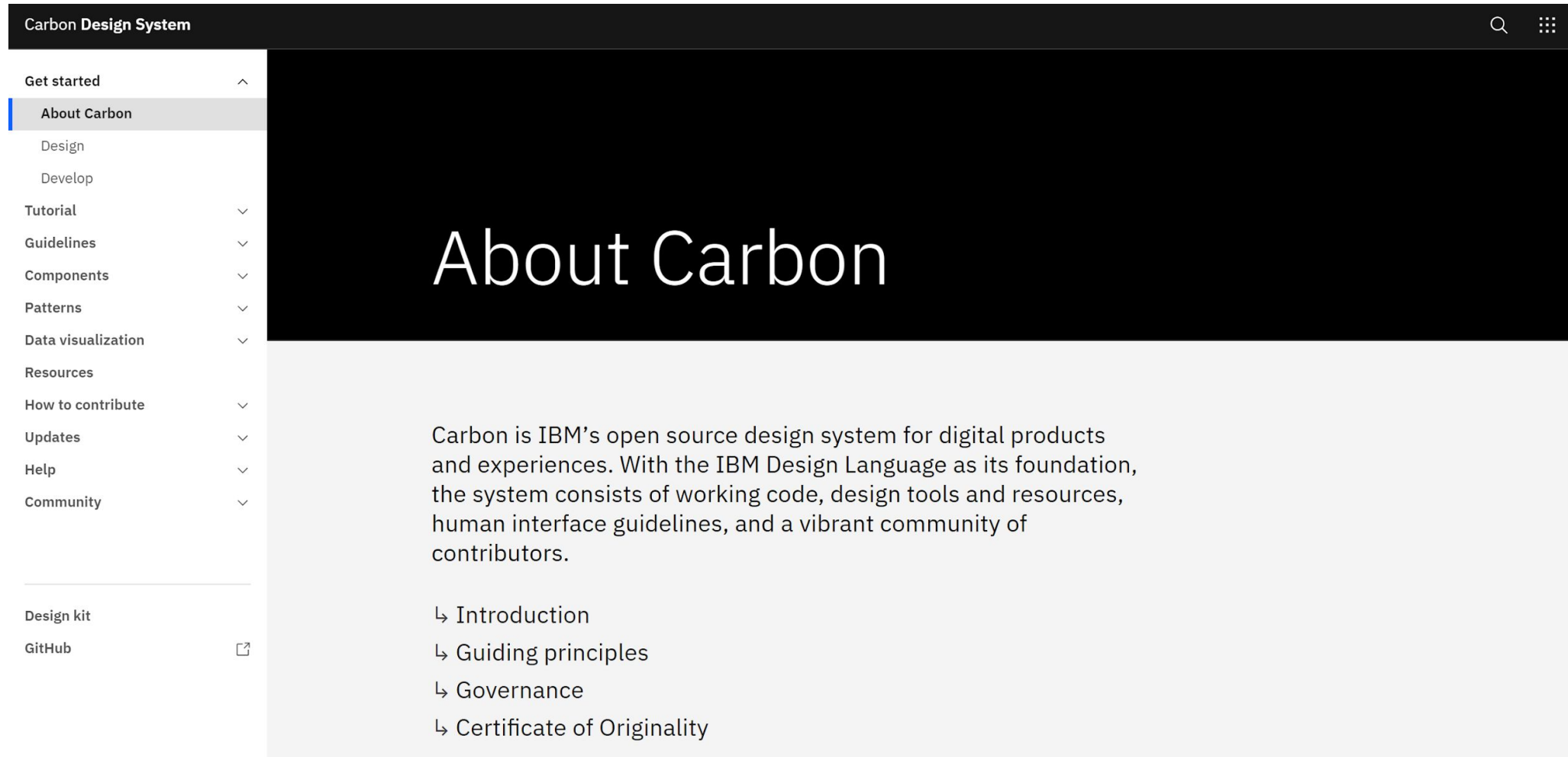
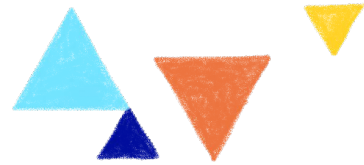
Design systems



Design System \approx
Style Guide + Component Catalog



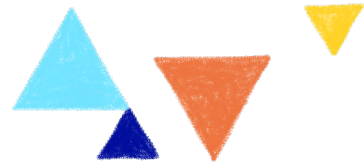
Example: Carbon Design System



<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

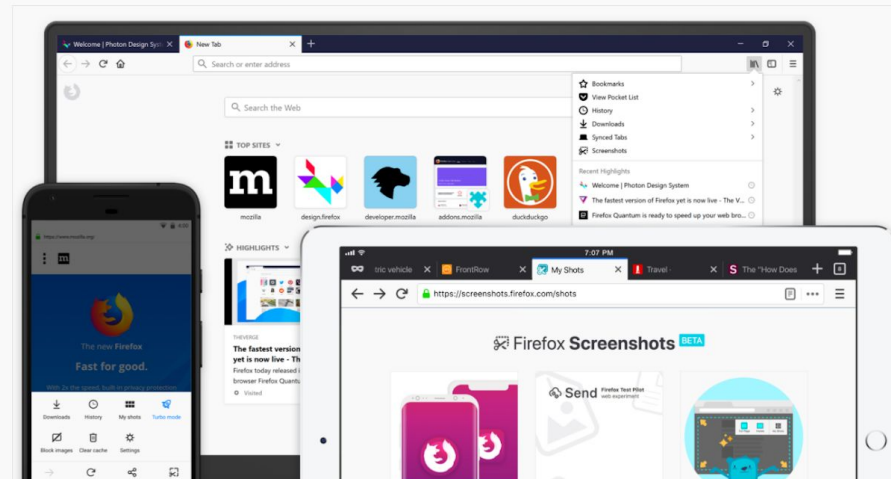
Patterns

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.

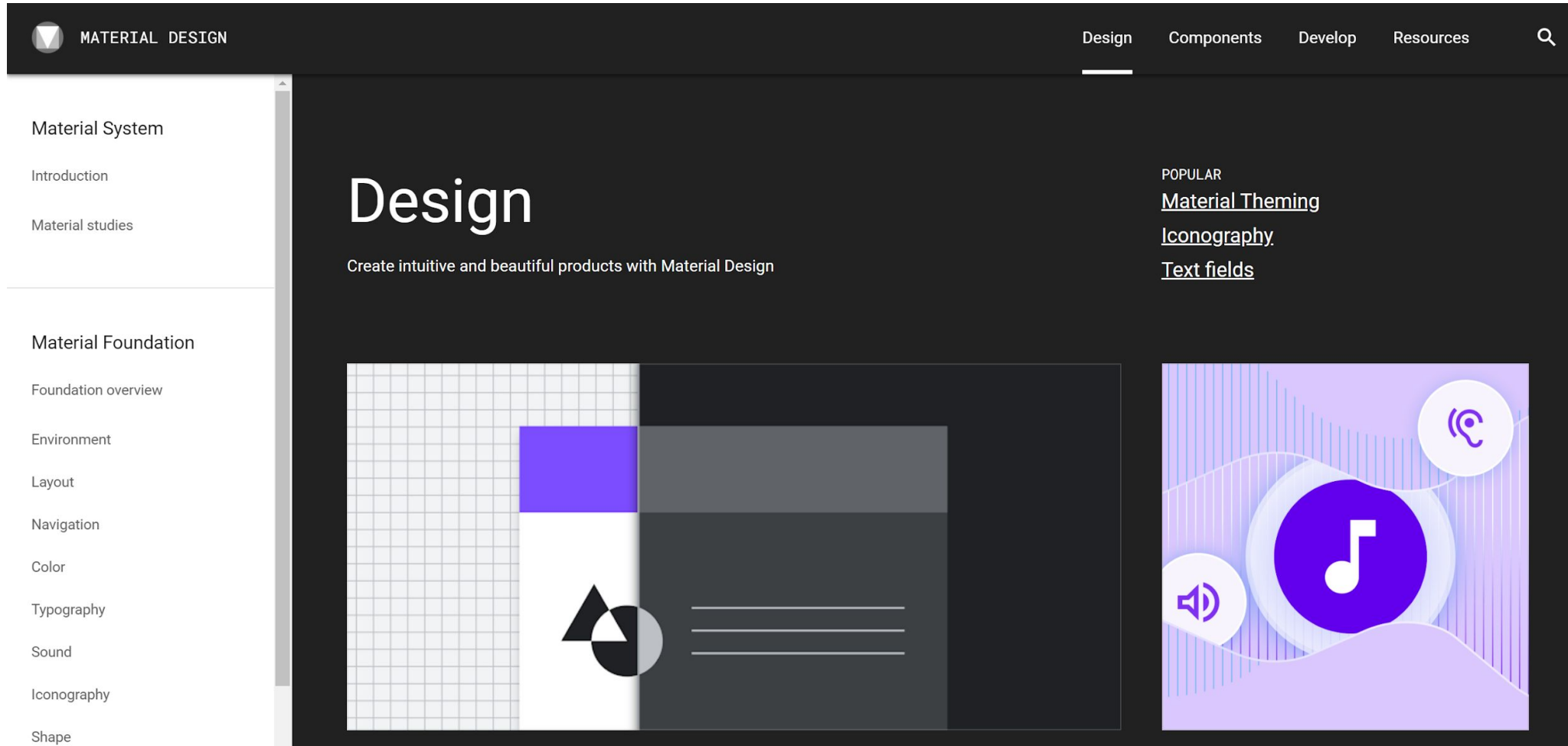
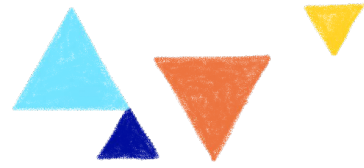
Using this system will help make your work more efficient, and our products more consistent, while still looking, feeling, and sounding uniquely Firefox.

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

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

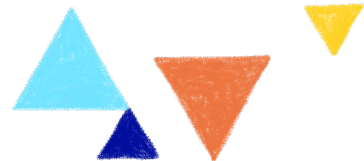


Example: Material Design



<https://material.io/>



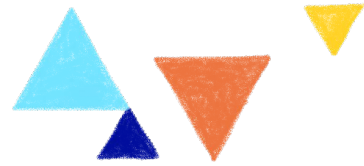


The component catalog

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!

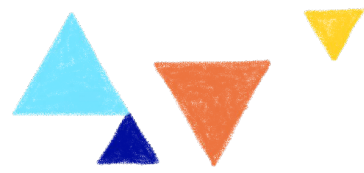
```
<div class="alert alert-primary" role="alert">
  A simple primary alert—check it out!
</div>
<div class="alert alert-secondary" role="alert">
  A simple secondary alert—check it out!
</div>
<div class="alert alert-success" role="alert">
  A simple success alert—check it out!
</div>
```



Bootstrap based component catalogs









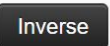

A long time ago



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
	<code>btn</code>	Standard gray button with gradient
	<code>btn btn-primary</code>	Provides extra visual weight and identifies the primary action in a set of buttons
	<code>btn btn-info</code>	Used as an alternative to the default styles
	<code>btn btn-success</code>	Indicates a successful or positive action
	<code>btn btn-warning</code>	Indicates caution should be taken with this action
	<code>btn btn-danger</code>	Indicates a dangerous or potentially negative action
	<code>btn btn-inverse</code>	Alternate dark gray button, not tied to a semantic action or use
	<code>btn btn-link</code>	Deemphasize a button by making it look like a link while maintaining button behavior

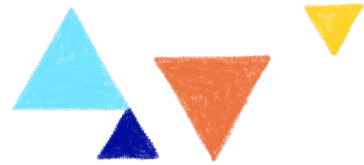


Bootstrap

Components defined in HTML, CSS and some jQuery



Then it was AngularJS time...



UI Bootstrap Directives ▾ Getting started Previous docs ▾

UI Bootstrap

Bootstrap components written in pure AngularJS by the AngularUI Team

[Code on Github](#) [Download \(2.5.0\)](#) [Create a Build](#)

[Star](#) 14,640 [Fork](#) 7,184 [Tweet](#)

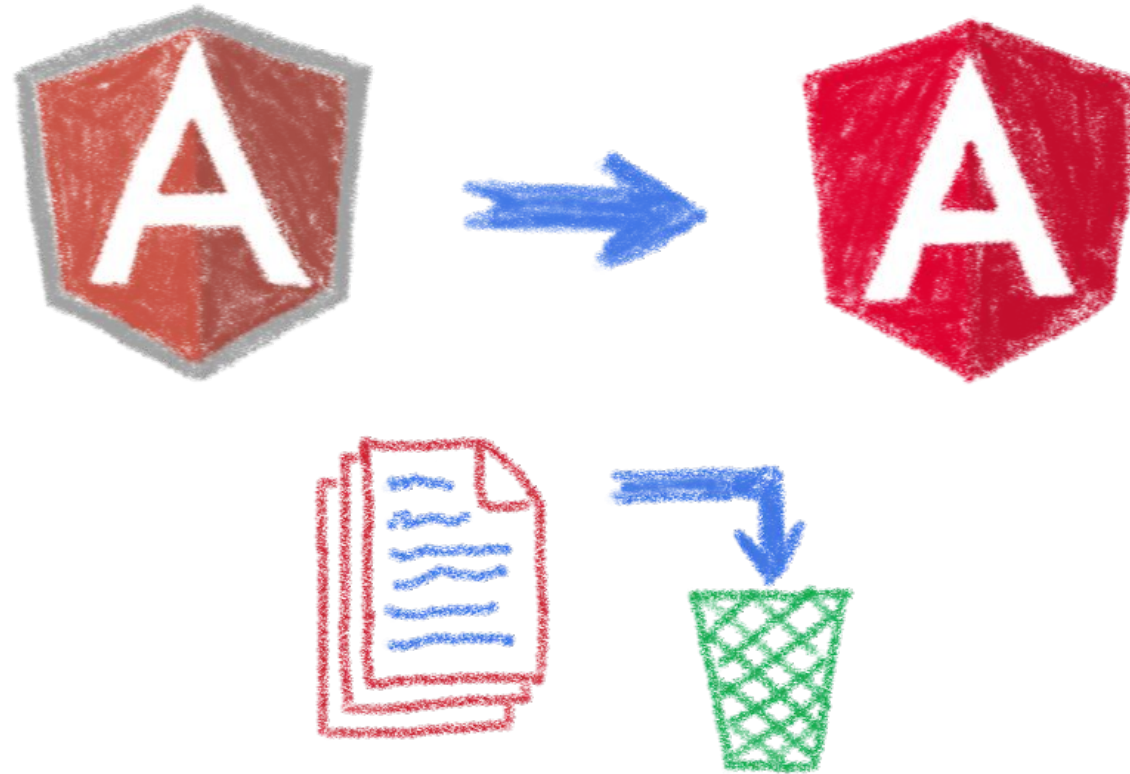
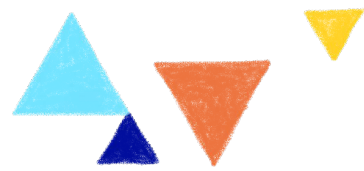


Getting started

And new reference implementations were needed



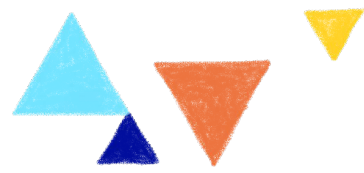
But you know the sad story...



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.

[Demo](#)[Installation](#)

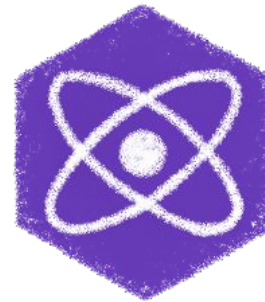
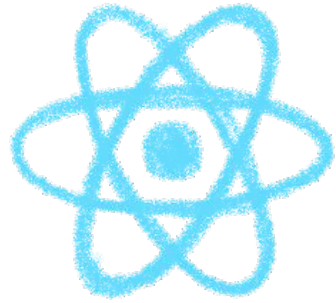
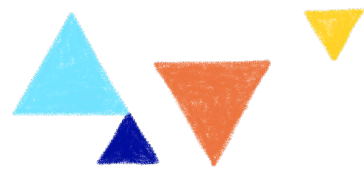
Currently at v6.1.0



ng-bootstrap to the rescue



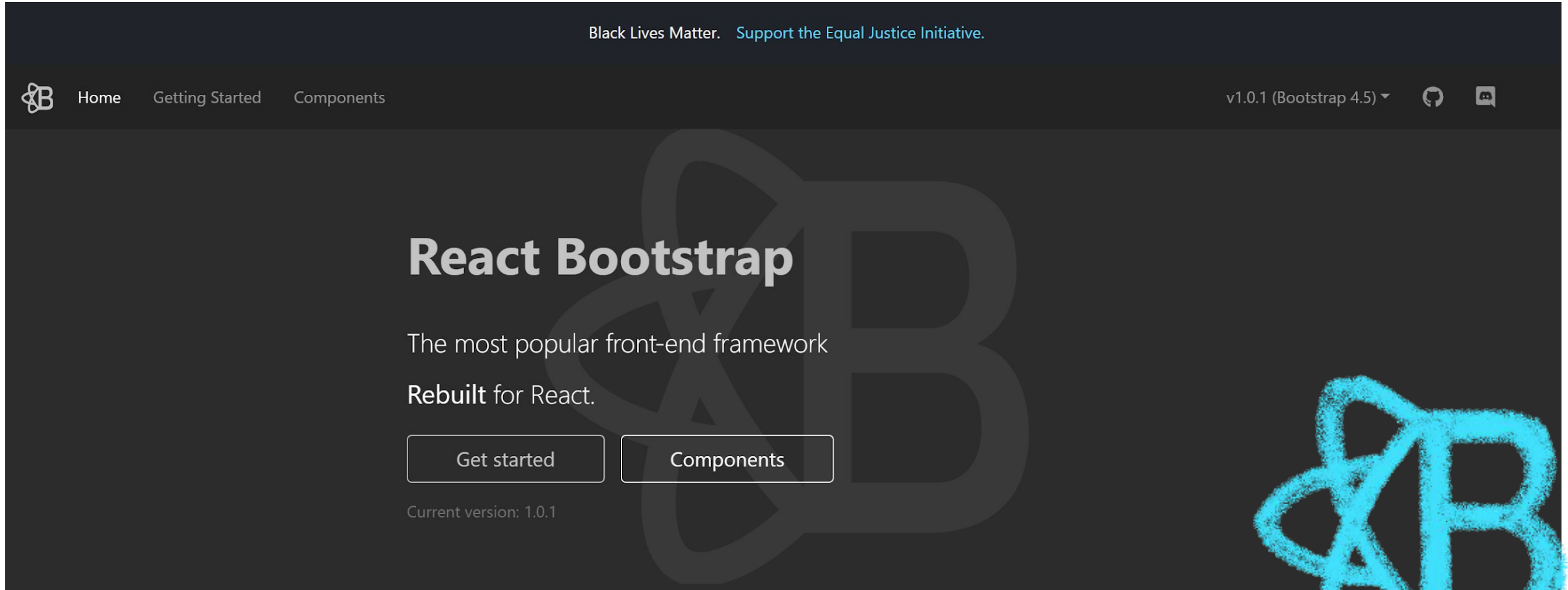
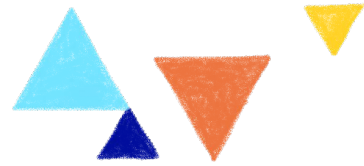
But times had changed...



In 2017 Angular is only one more in the clique



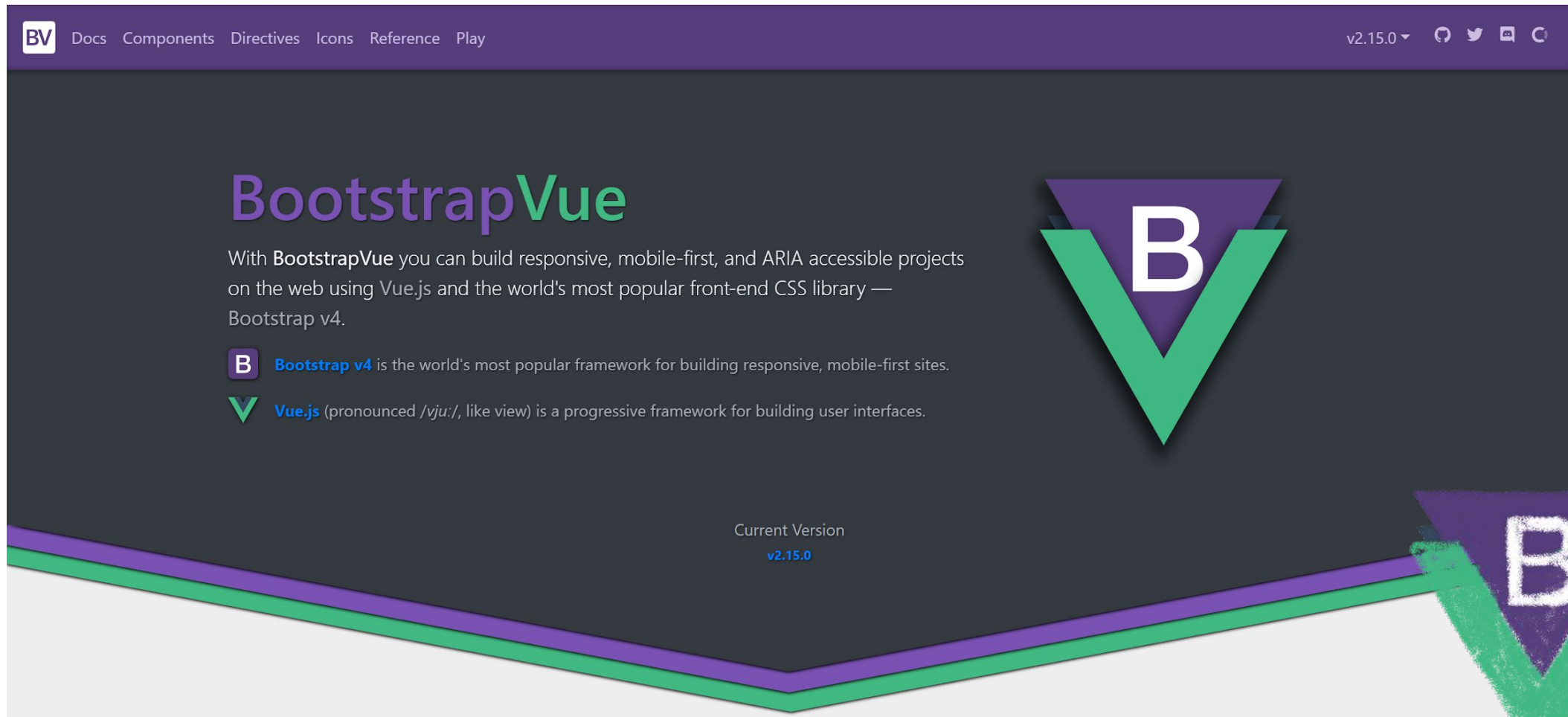
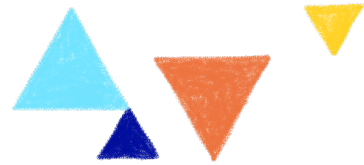
React is the new Big Thing™



So let's build React Bootstrap...



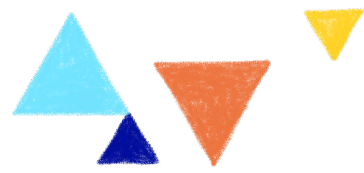
Wait, what about Vue?



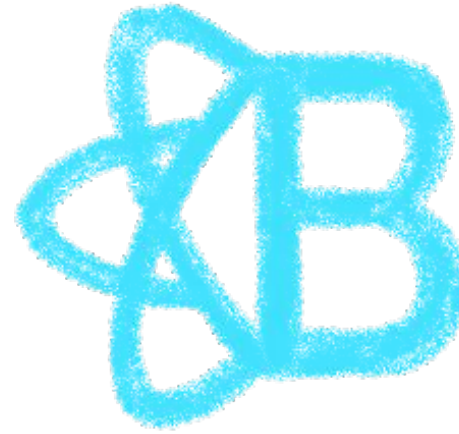
We also need BootstrapVue



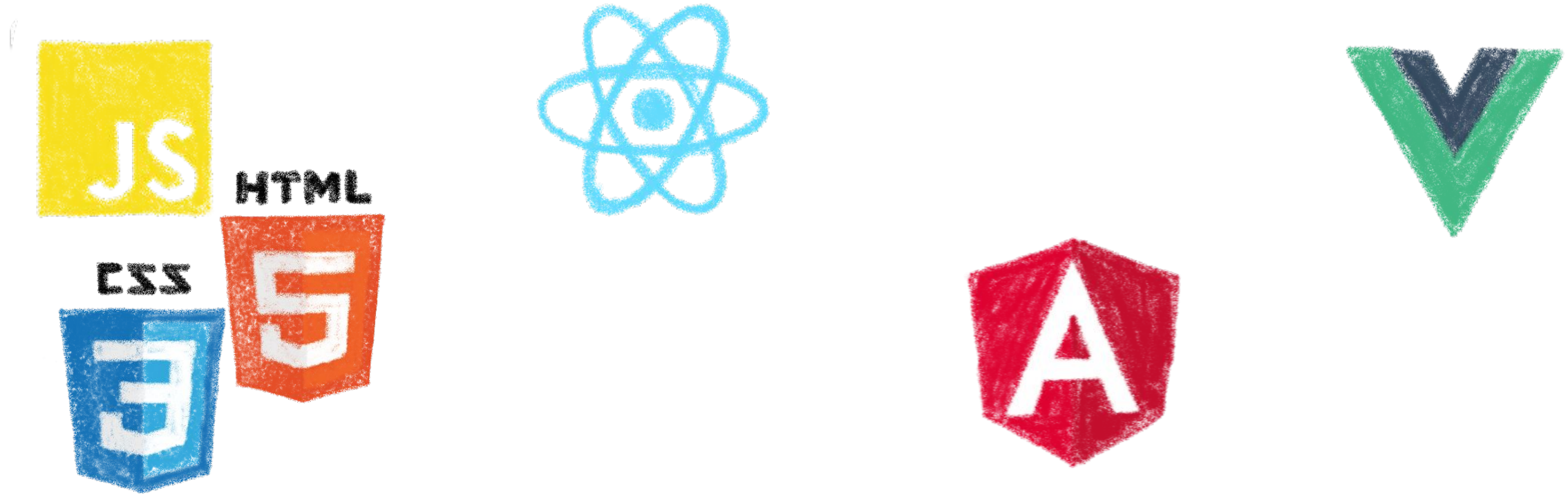
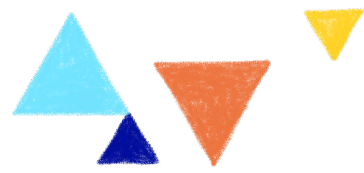
OK, I think you see my point...



Bootstrap



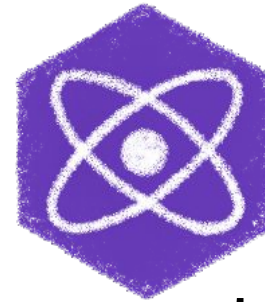
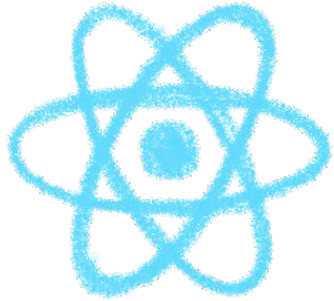
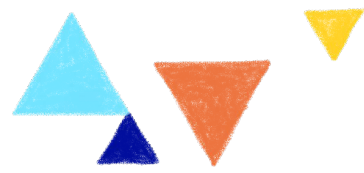
Most Design System do a choice



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



Both choices are problematic

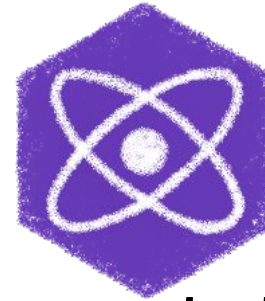
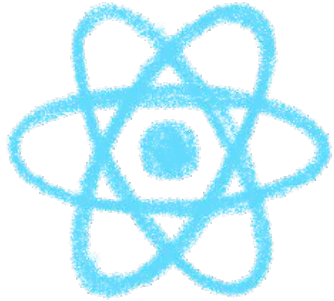
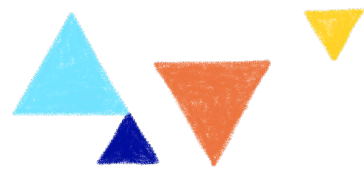


Shipping only one implementation:

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



Both choices are problematic

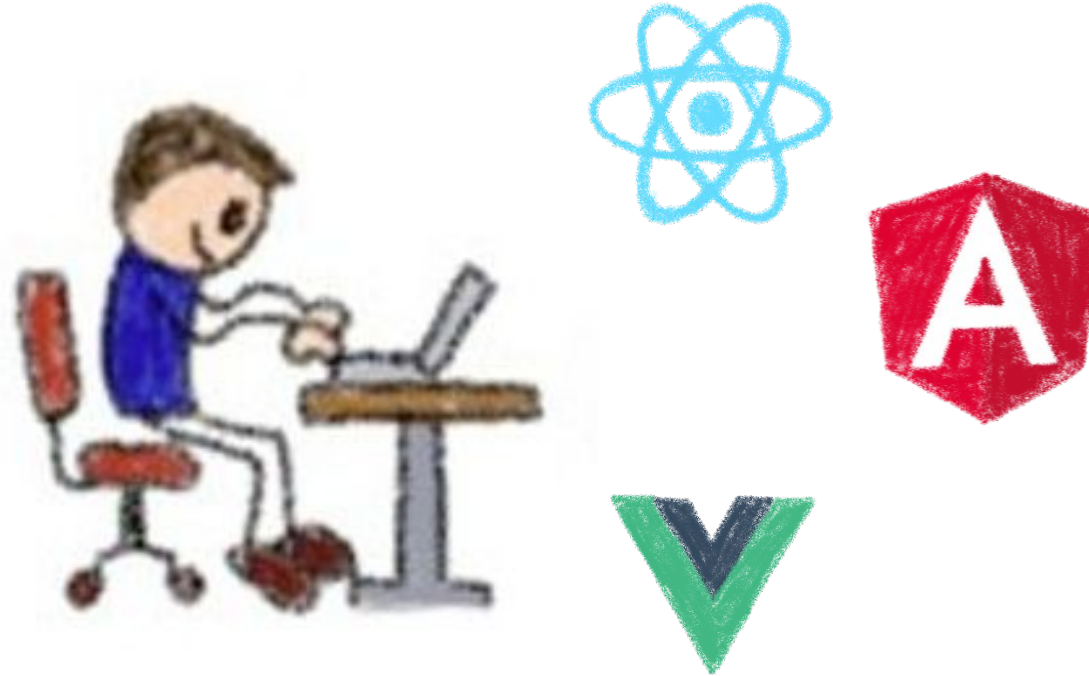
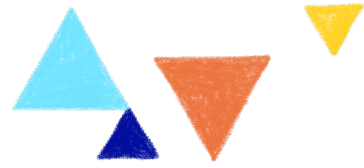


Shipping several implementations:

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



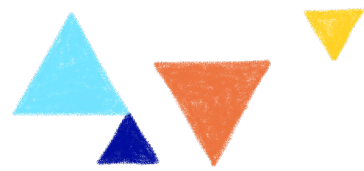
Incomplete catalogs are problematic



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



Example: Carbon Design System



Carbon Design System

Get started

About Carbon

Design

Develop

Tutorial

Guidelines

Components

Patterns

Data visualization

Resources

How to contribute

Updates

Help

Community

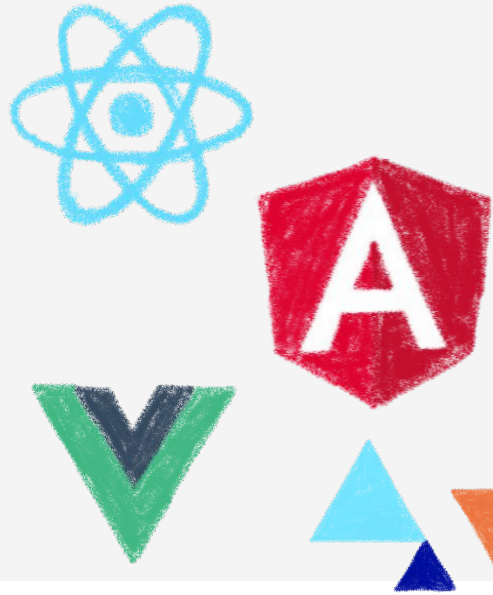
Design kit

GitHub

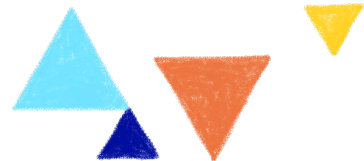
About Carbon

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





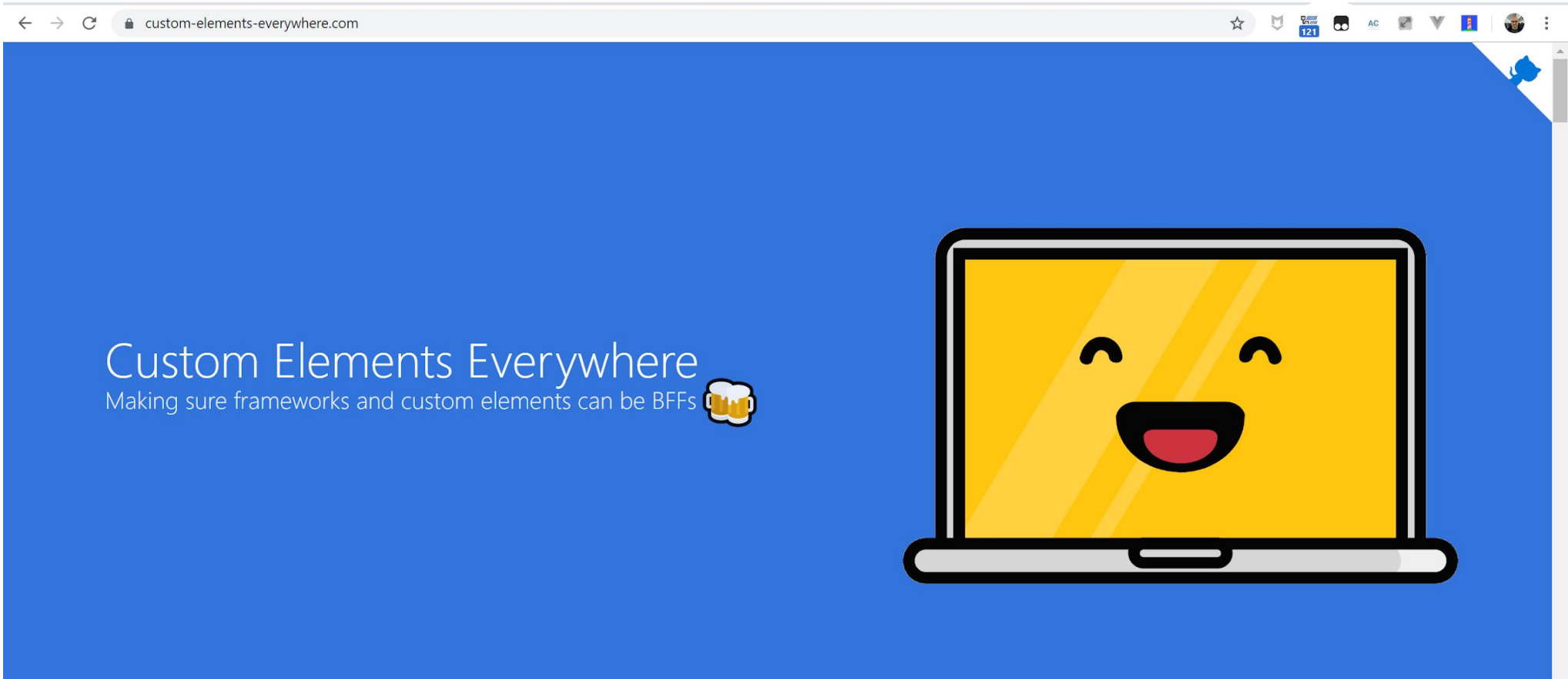
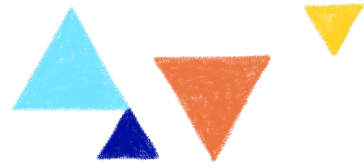


Web Components & Design Systems

A match made in heaven



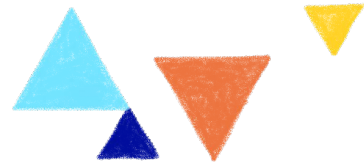
Compatibility is on Web Components side



Web Components everywhere, baby!



They are truly everywhere



↑ **spacexfsw**  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

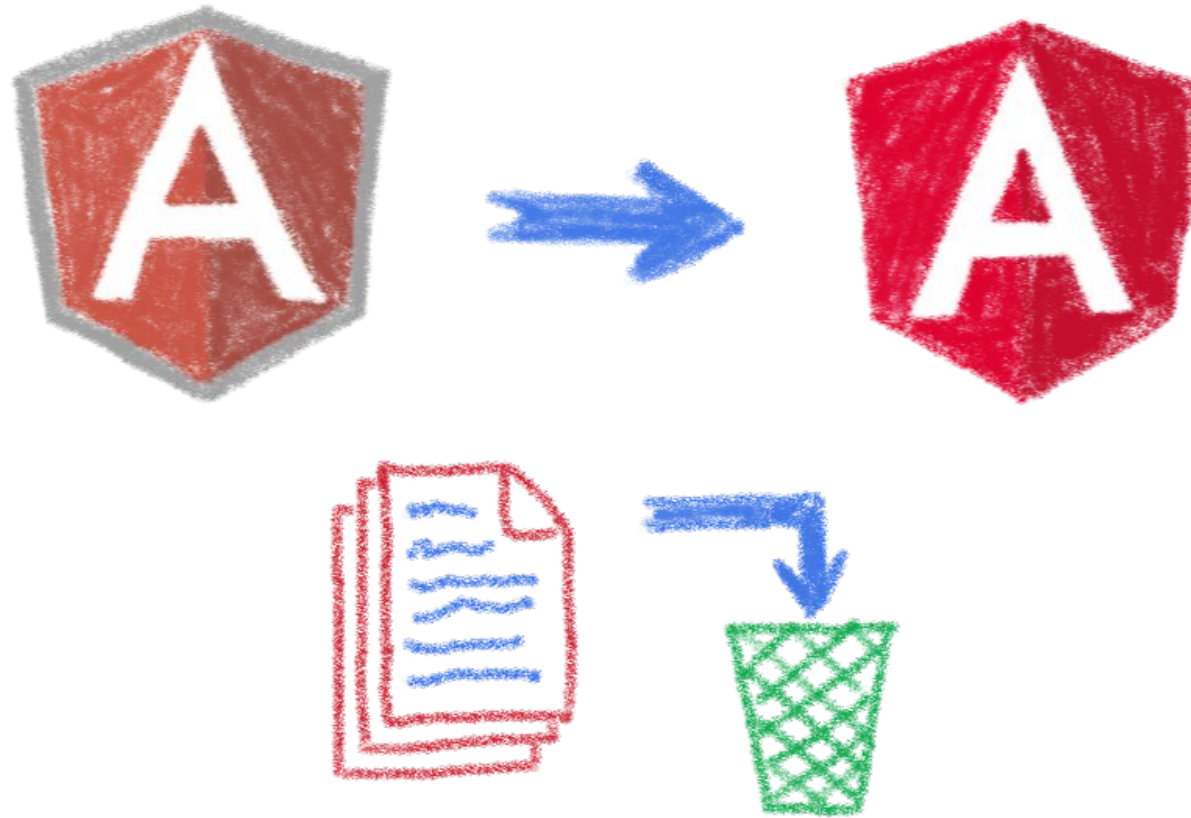
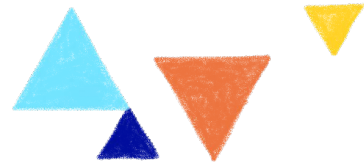
Different team members uses different editors, I use VSCode but I might be just a little bit biased :) - Sofian

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 spaaaaaaace 



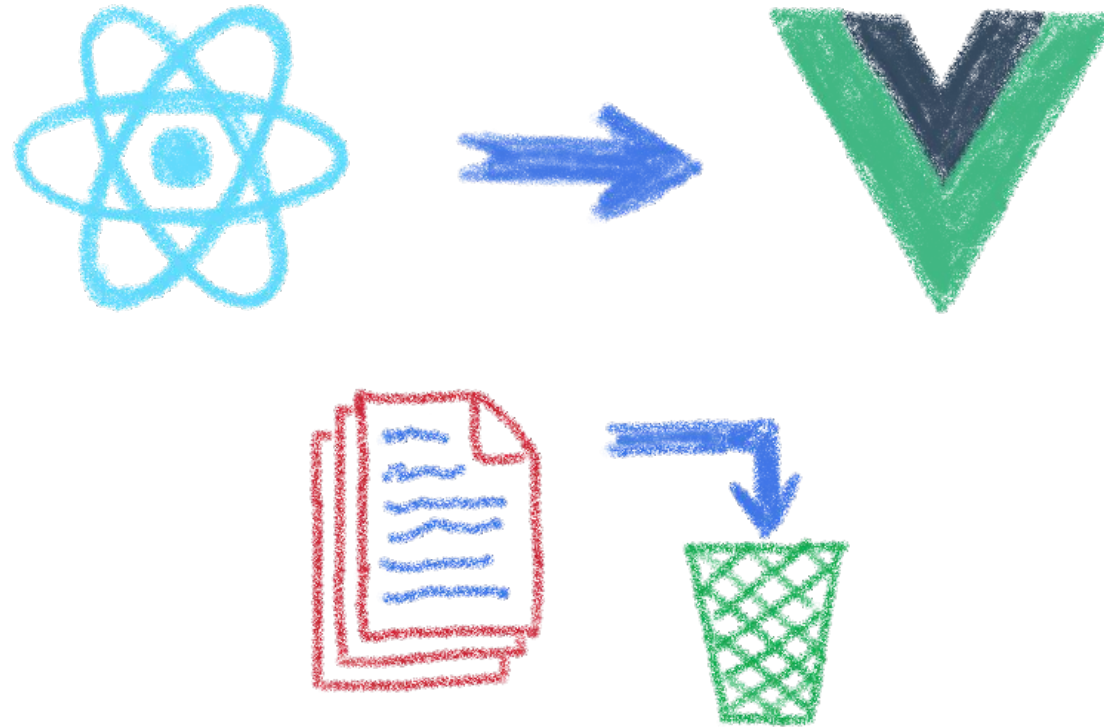
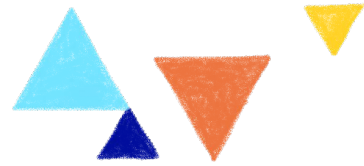
Do you remember AngularJS?



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



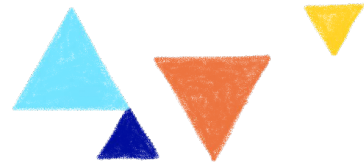
The pain of switching frameworks?



Rewriting once again your code...



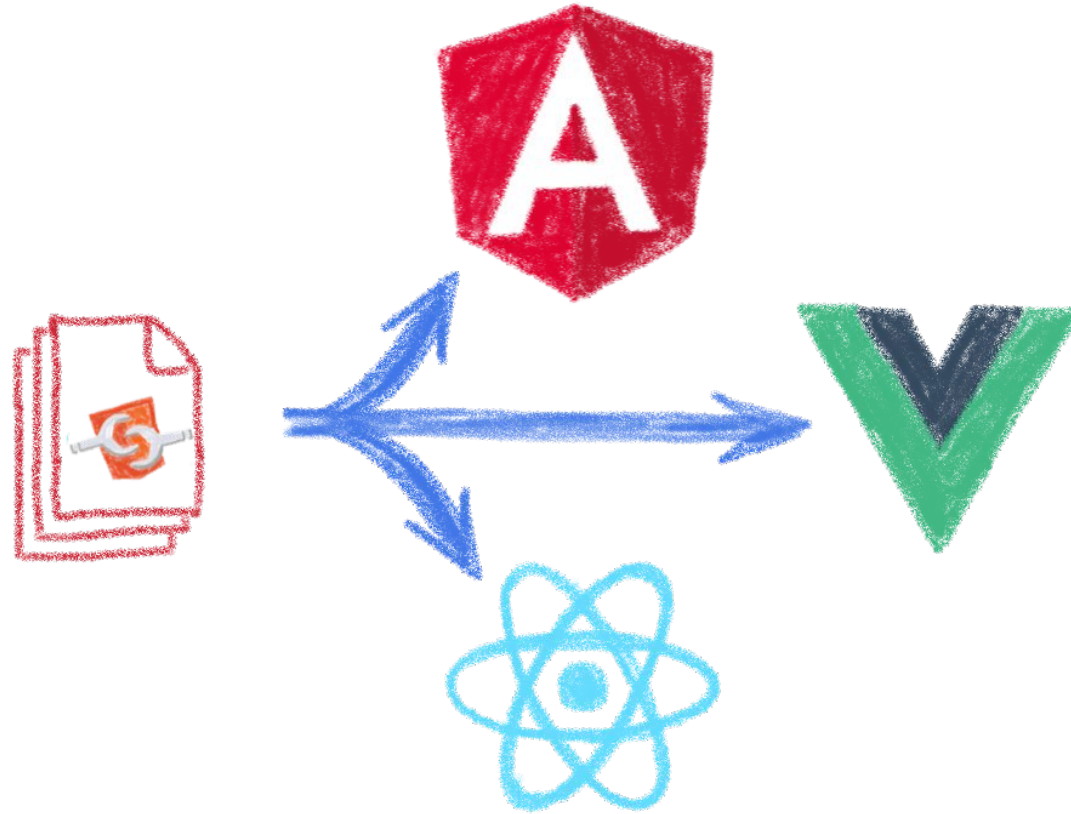
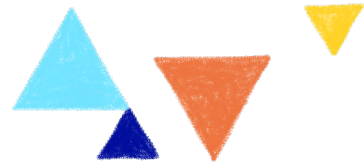
The impossibility of sharing UI code?



Between apps written with different frameworks



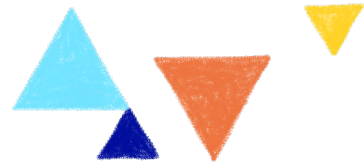
Web Components change that



In a clean and standard way



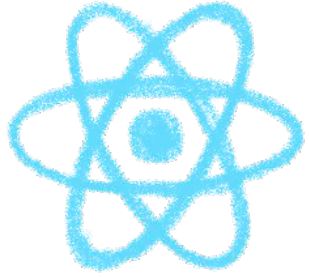
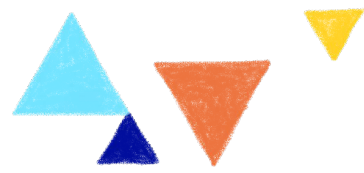
They are the interoperable alternative



Any framework... or no framework at all



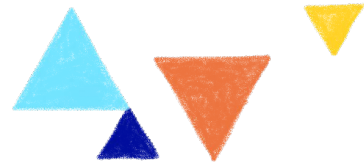
You can have a single implementation



And it simply works everywhere

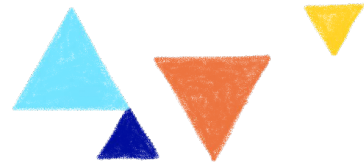


When you need interoperability



Nothing beats the standard



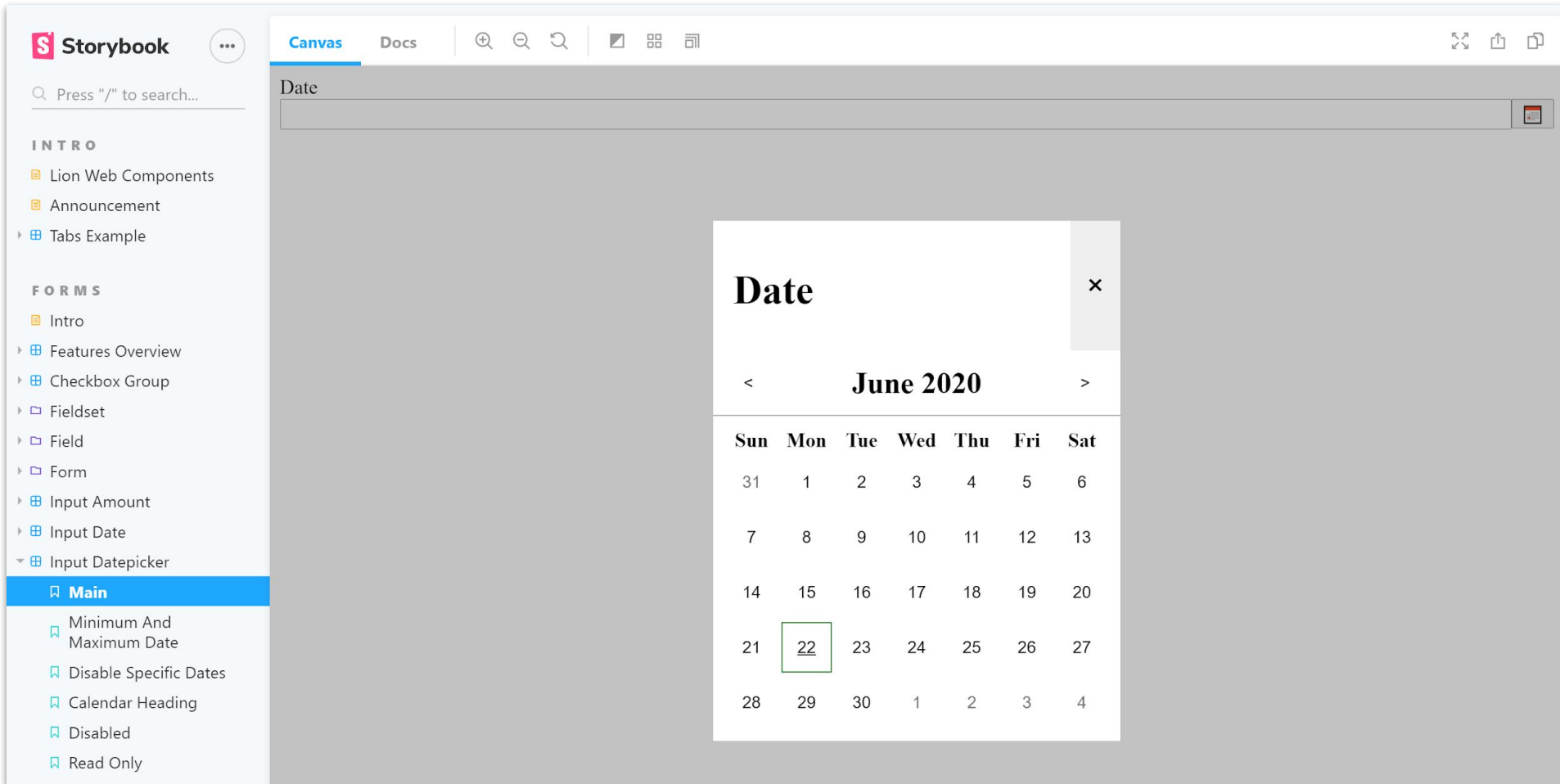
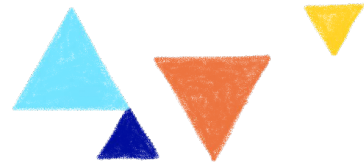


But how to do it?

**Designing, developing and managing
a catalog of Web Components**



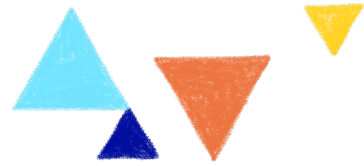
Learning from the best



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



Learning from the best



The screenshot shows the 'clever cloud' documentation page for 'Collection of Web Components by Clever Cloud'. The left sidebar contains a navigation menu with sections: HOME (Readme, Changelog, Contributing, Release), DOCS (Architecture Decision Records, How to translate and localize?, How to write stories, Web components guidelines), and ATOMS (a list of component tags like <cc-beta>, <cc-button>, etc.). The main content area has a title 'Collection of Web Components by Clever Cloud', followed by a section 'What is this?' which explains the project's purpose and lists some low-level components like <cc-button>, <cc-input-text>, and <cc-loader>. Below this is a section 'Why is it public?' with three numbered points explaining the motivation for sharing the components.

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.

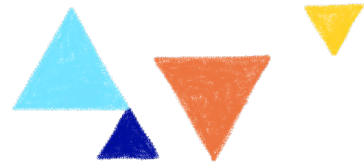
We use them on different Web UIs we have (public and internal).

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.

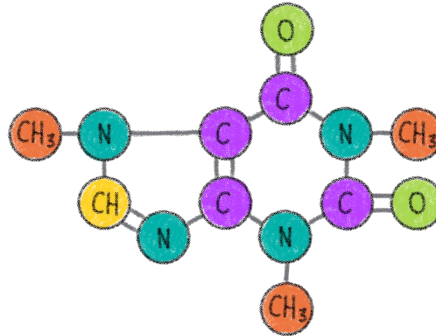
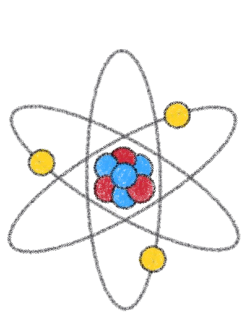
<https://github.com/CleverCloud/clever-components>



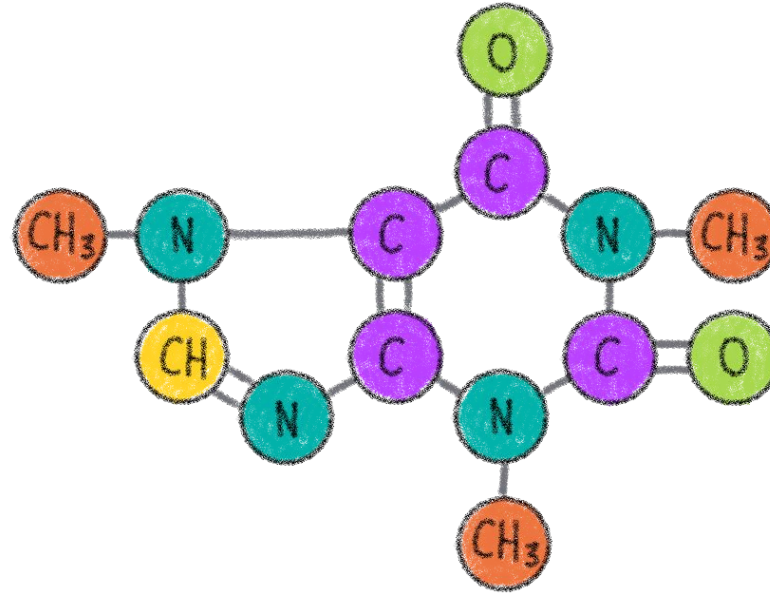
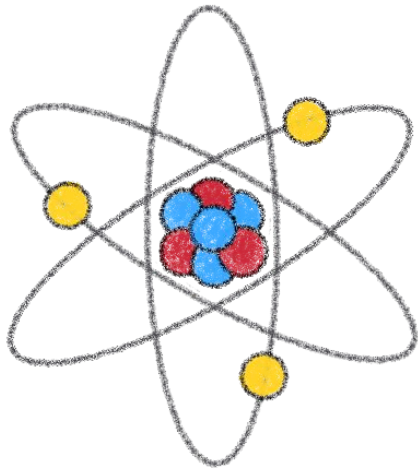
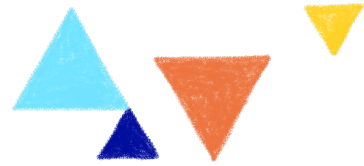


What kind of components?

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



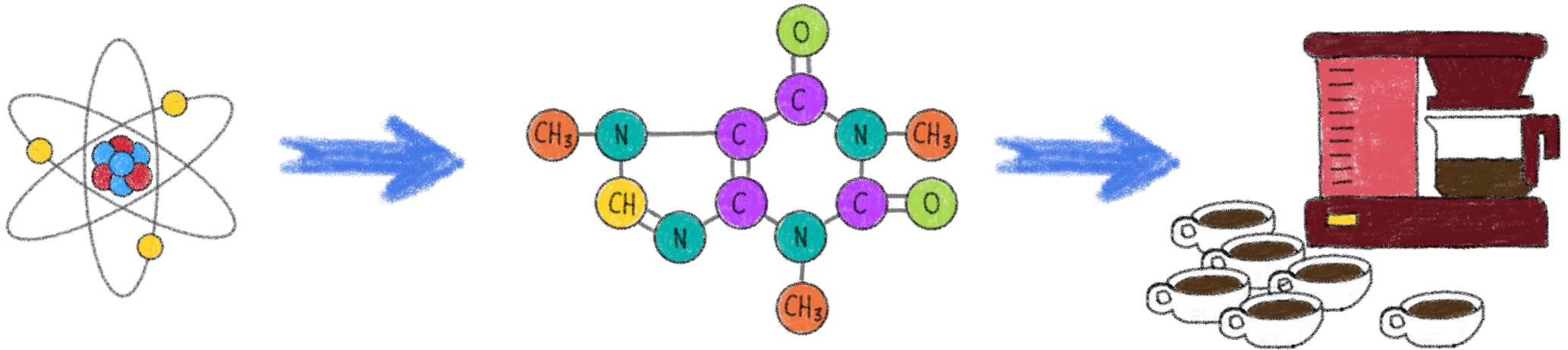
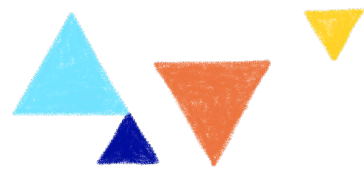
A matter of size and complexity



What kind(s) of components you want to build



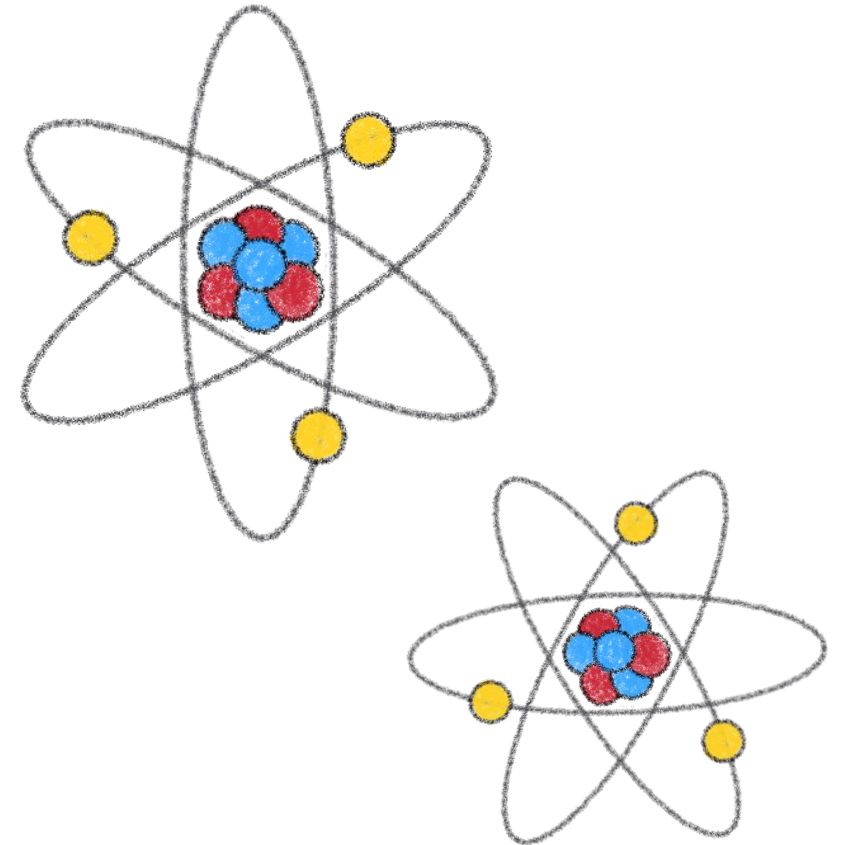
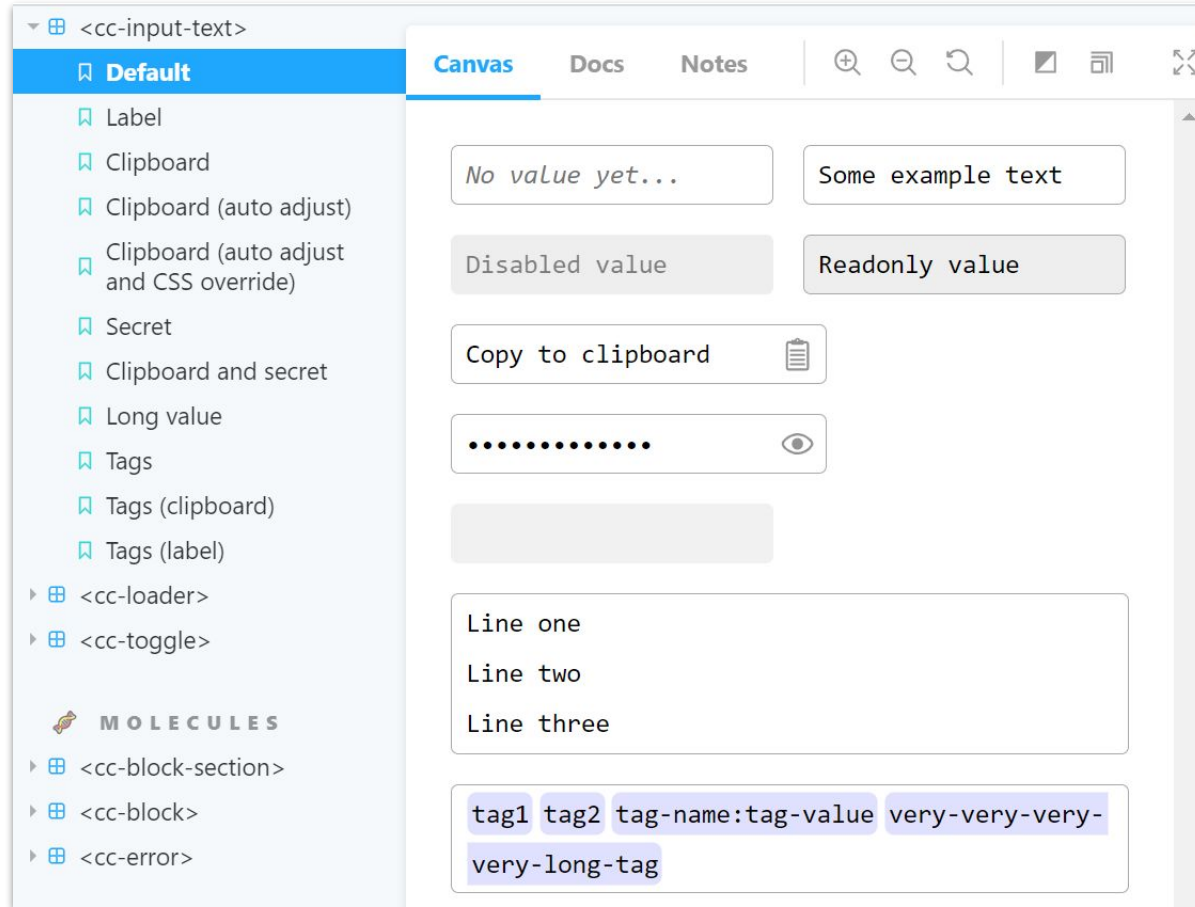
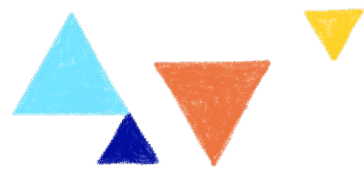
Build from the bottom and go up



Eat your own dog food



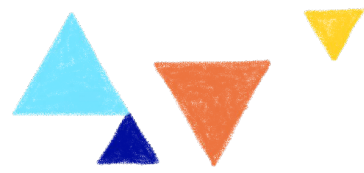
And how to choose the atoms?



Flexibility and configurability are key



And how to choose the atoms?



Input Iban

Main

Prefilled

Faulty Prefilled

Country Restrictions

Input Range

Input Stepper

Input

Listbox

Radio Group

Select Rich

Select

Textarea

ValidateMessages

Validation

System

BUTTONS

Intro

Button

Switch

OVERLAYS

Intro

Dialog

Tooltip

Scope

System

Canvas

Docs

Input IBAN

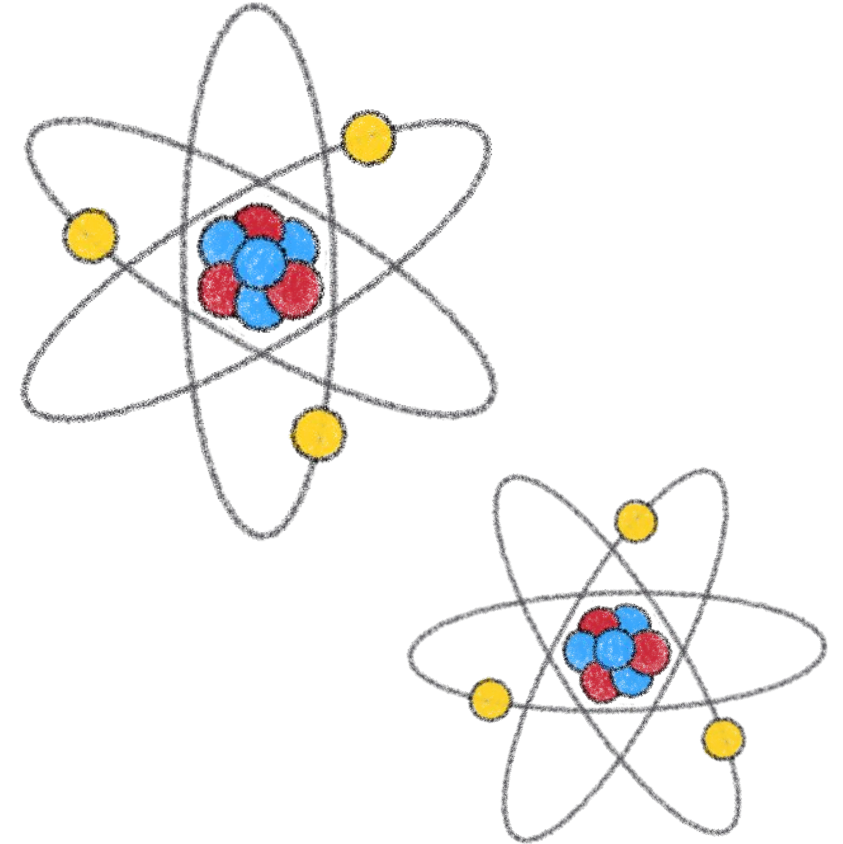
`lion-input-iban` component is based on the generic text input field. Its purpose is to provide a way for users to fill in an IBAN (International Bank Account Number).

Account

Show code

Features

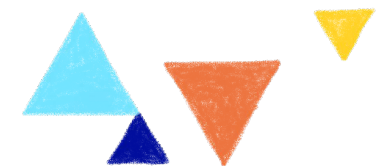
- Based on [lion-input](#)
- Default label in different languages
- Makes use of IBAN specific [validate](#) with corresponding error messages in different languages
 - IsIBAN (default)
 - IsCountryIBAN
- Parses IBANs automatically
- Formats IBANs automatically



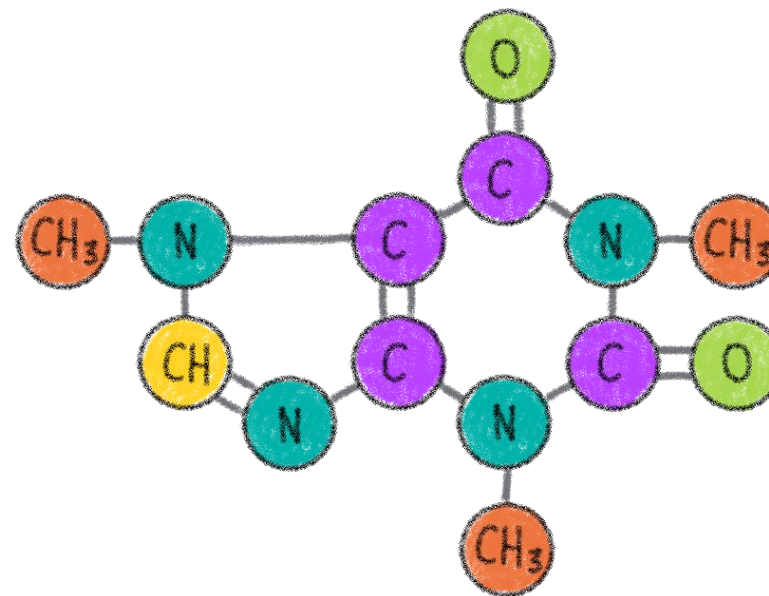
Encode often used patterns



And what about the molecules?



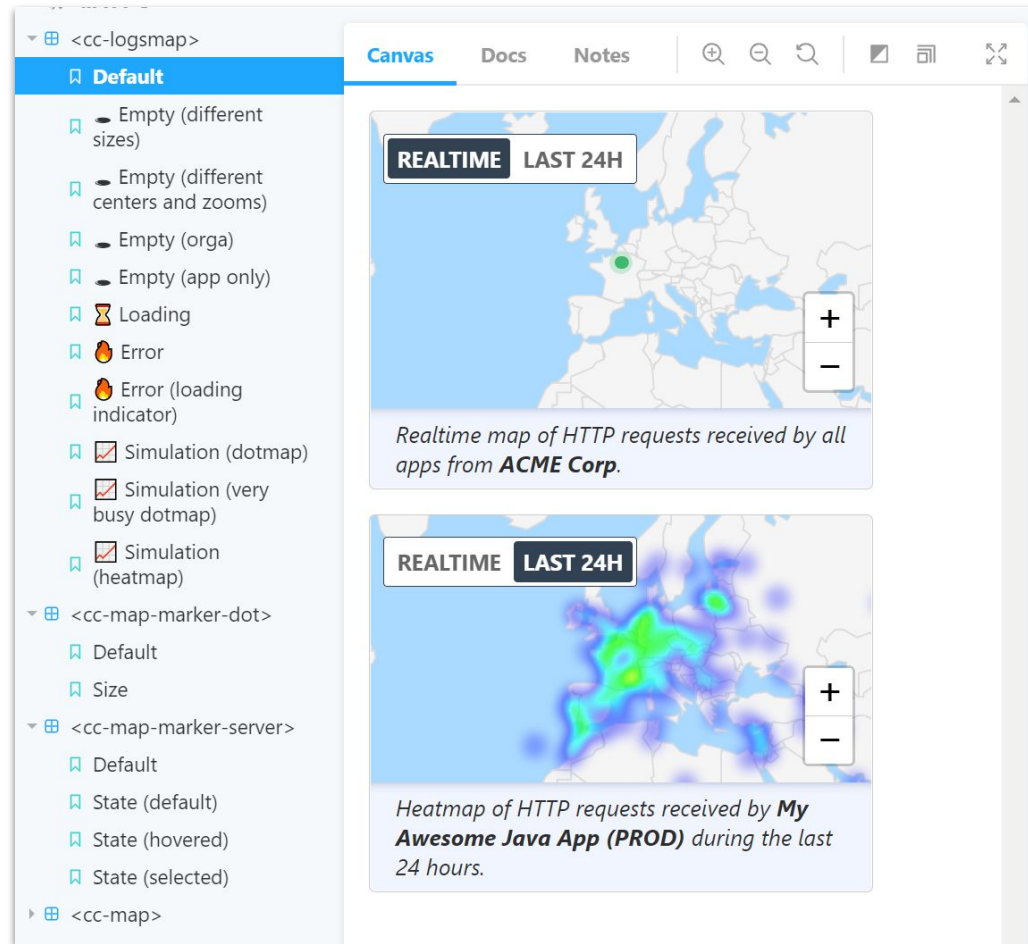
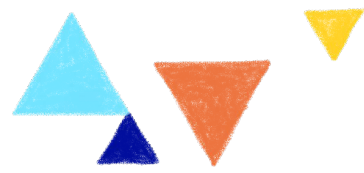
The screenshot shows a web editor interface. On the left is a sidebar with a list of blocks: 'Clipboard and secret', 'Long value', 'Tags', 'Tags (clipboard)', 'Tags (label)', '<cc-loader>', '<cc-toggle>', a 'MOLECULES' section with an icon, '<cc-block-section>', '<cc-block>' (expanded), 'Default', 'Overlay (loader)', 'Overlay (error alert)', 'Icon' (highlighted), 'Button', 'State (open)', 'State (close)', 'State (overflow)', 'Icon and open', and '<cc-error>'. The main canvas area has tabs for 'Canvas', 'Docs', and 'Notes'. The 'Canvas' tab is active, showing a block titled 'This is my block' with a green icon. Below the title is a paragraph of Lorem ipsum text. The 'Notes' tab is also visible, showing a paragraph of text.



Capitalize on your atoms
Keep the flexibility and configurability



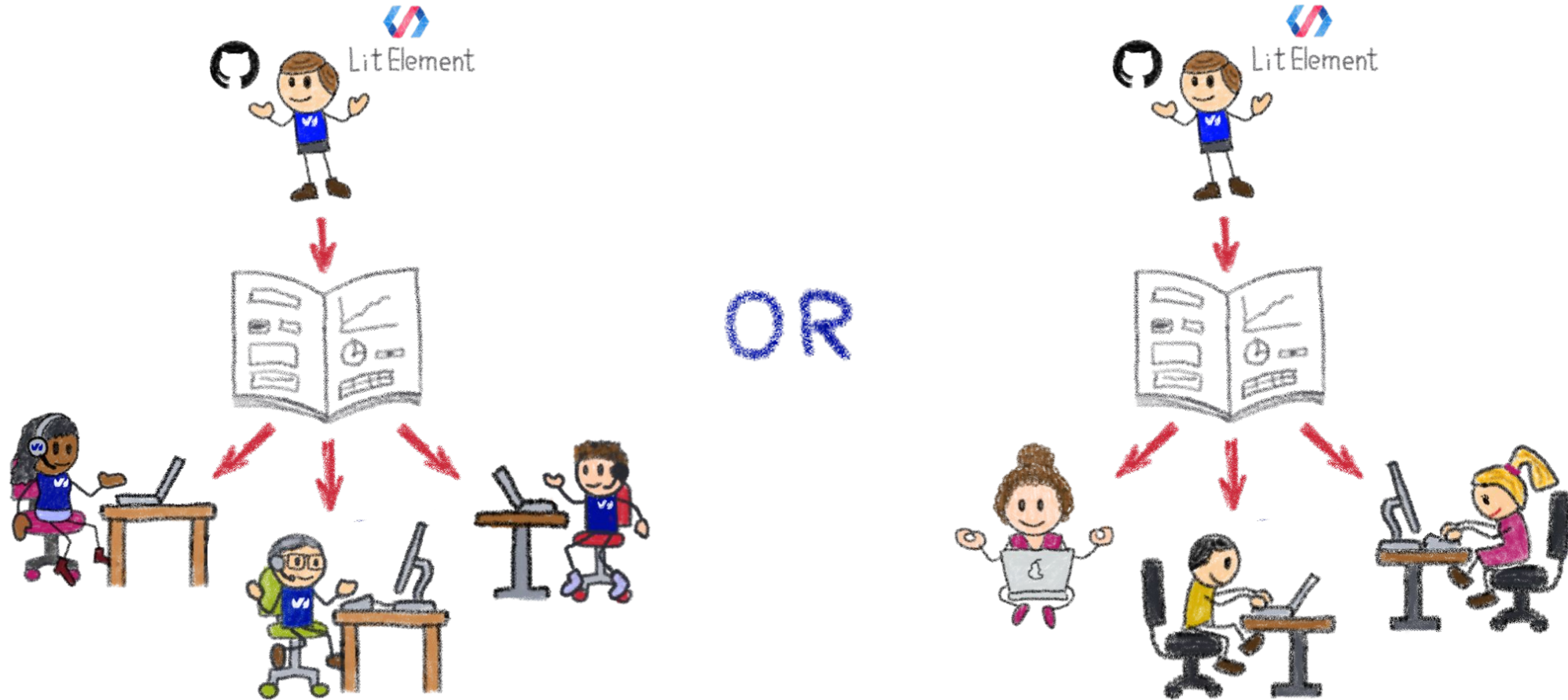
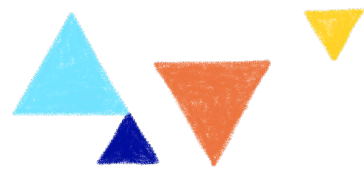
Big smart business components



Encoding your business logic



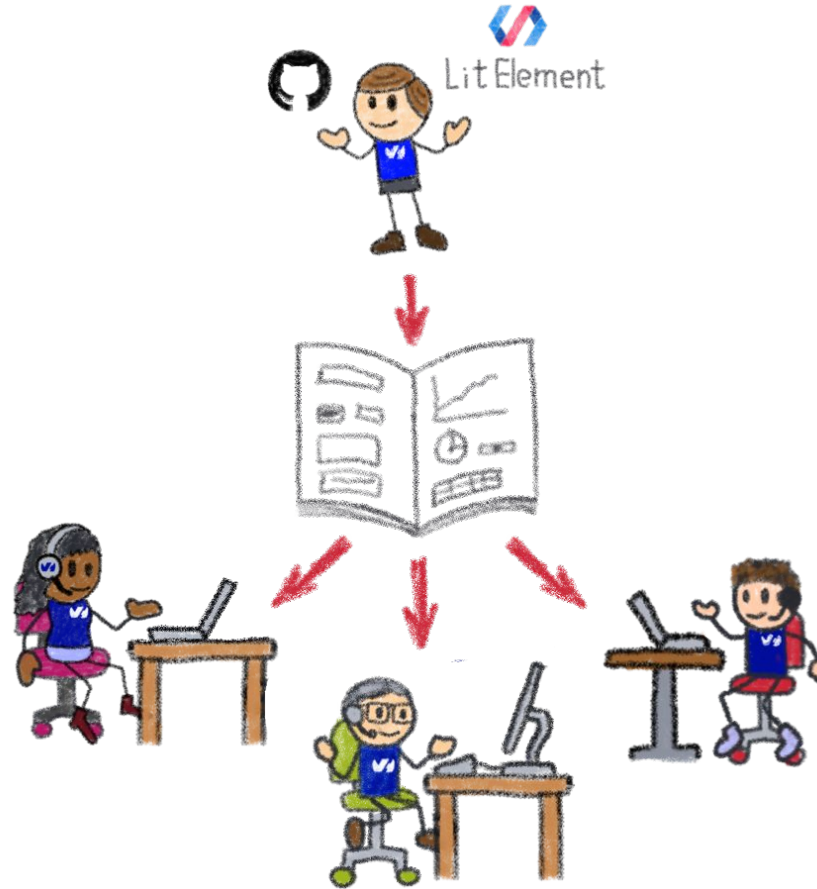
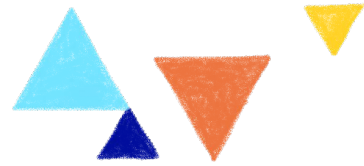
Internal or external customers?



Who are your target users?



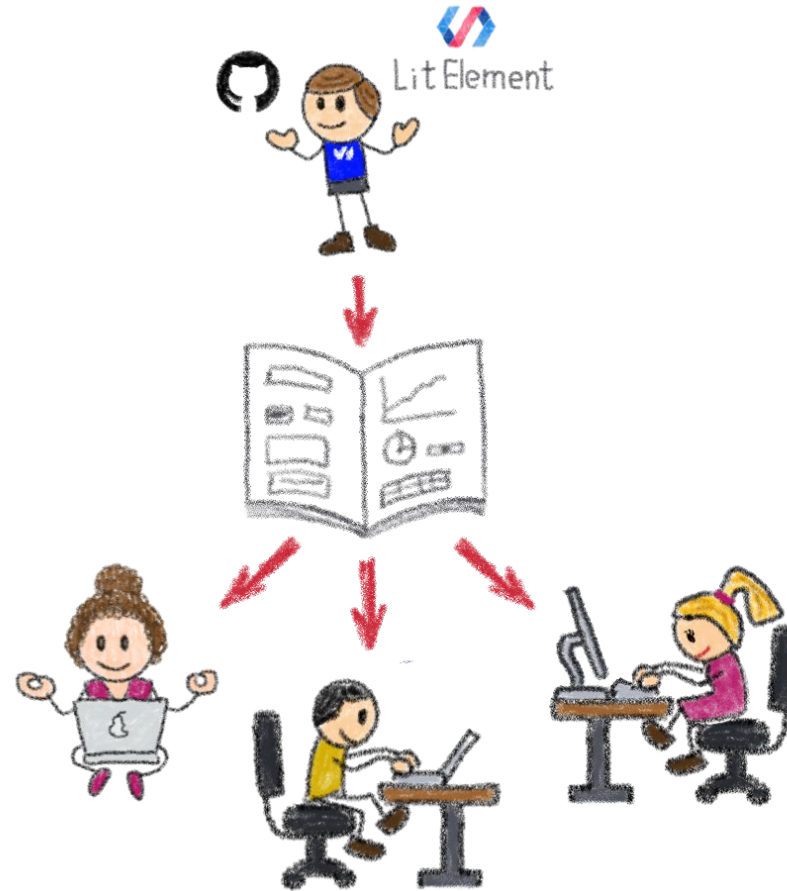
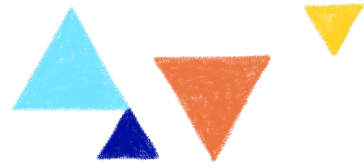
Internal customers need off-the-shelf components



A well defined and coherent look-and-feel

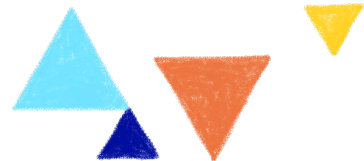


External customers need to be able to tweak



Theming and customizing components



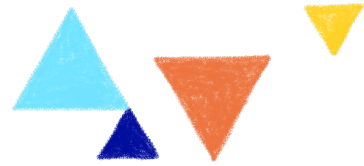


How to organize the catalog

Packages, imports and pragmatism



A single repository



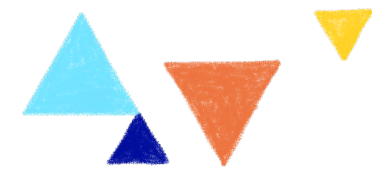
The screenshot shows the GitHub interface for the repository 'ing-bank / lion'. The repository has 28 watches, 887 stars, and 144 forks. The 'Code' tab is selected, showing a list of files in the 'lion / packages /' directory. The files are listed with their names, commit messages, and the time since the last commit.

File Name	Commit Message	Last Commit
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	✓

Noisy Pop MusicProductsPricingDocumentationCommunity

npmSearch packagesSearchSign UpSign In

Learn about our RFC process, Open RFC meetings & more. [Join in the discussion!](#)

@clevercloud/components4.1.2 • Public • Published 2 months ago

ReadmeExploreBETA10 Dependencies0 Dependents76 Versions

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

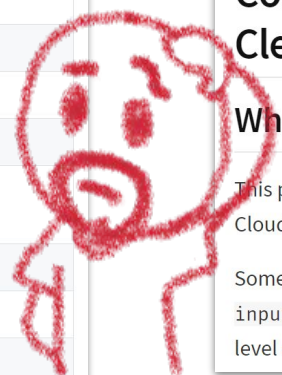
232

VersionLicense

4.1.2Apache-2.0

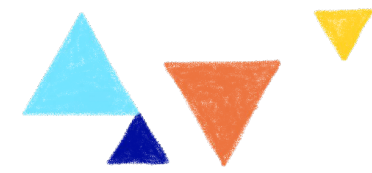
Unpacked SizeTotal Files

1.31 MB145



A packet per component or a global one

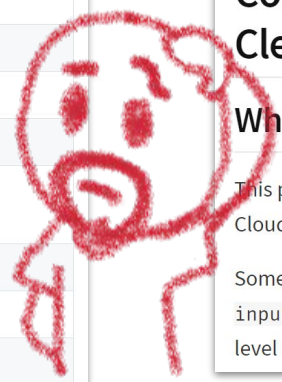
Two schools of thought



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	✓



Noisy Pop MusicProductsPricingDocumentationCommunity

npmSearch packagesSearchSign UpSign In

Learn about our RFC process, Open RFC meetings & more. [Join in the discussion!](#)

@clevercloud/components4.1.2 • Public • Published 2 months ago

ReadmeExploreBETA10 Dependencies0 Dependents76 Versions

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

VersionLicense

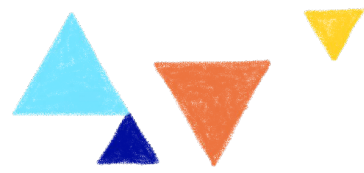
4.1.2Apache-2.0

Unpacked SizeTotal Files

1.31 MB145

Individual versioning vs global one

Lots of web components libraries



LitElement



hybrids



snuggsi ツ



SkateJS

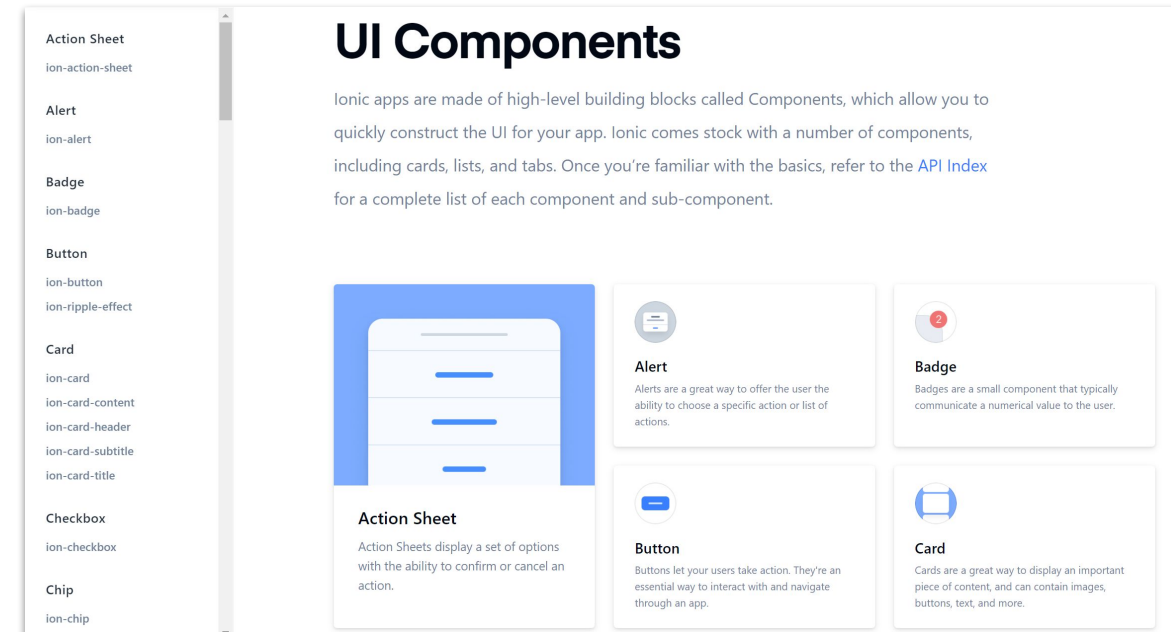
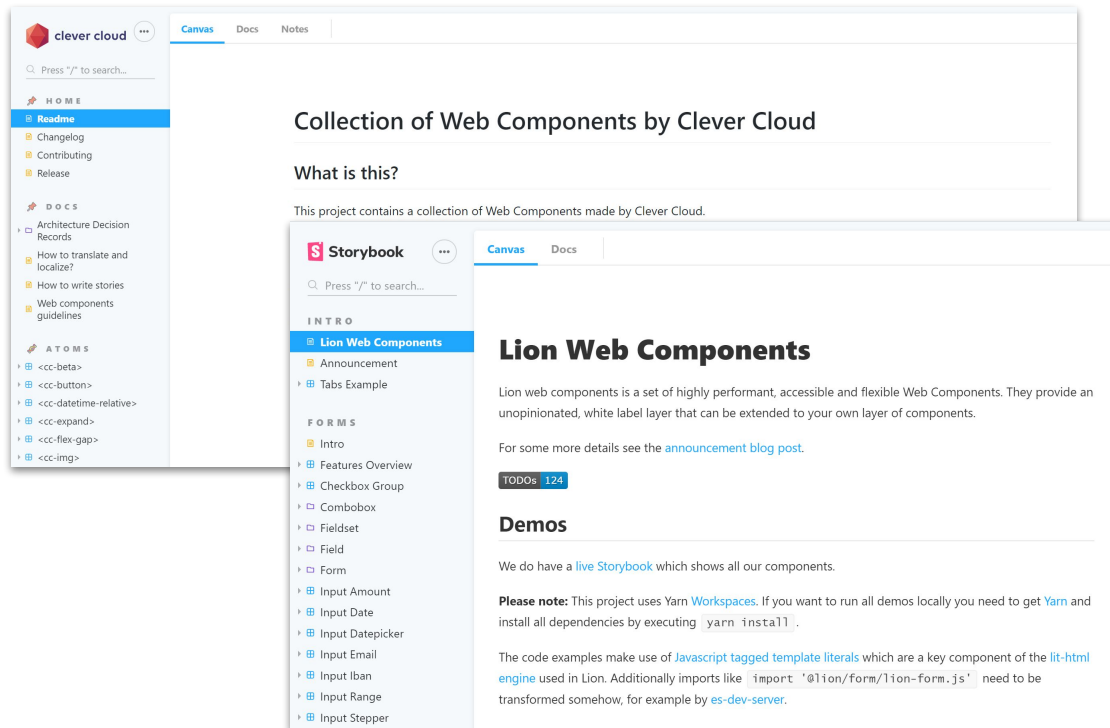
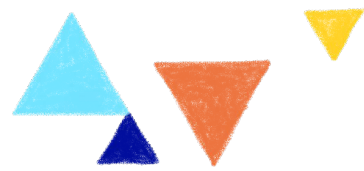


smart
HTML ELEMENTS

For different needs and sensibilities

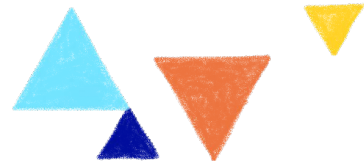


Which ones to use?



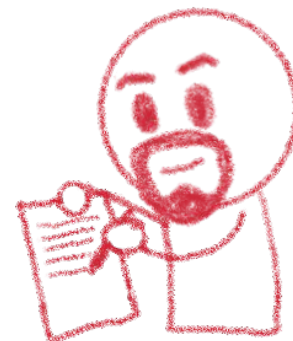
All are good, but these are popular favorites



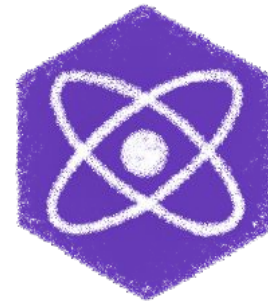
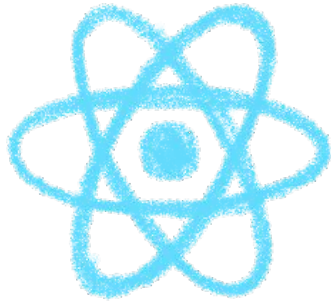
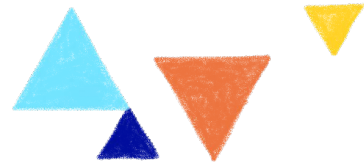


Driving-up adoption

Making devs use your components



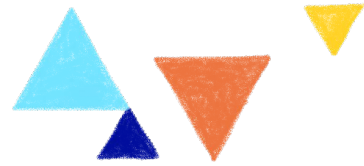
Think who are your target users



Users of any framework current or future...



They aren't used to your library



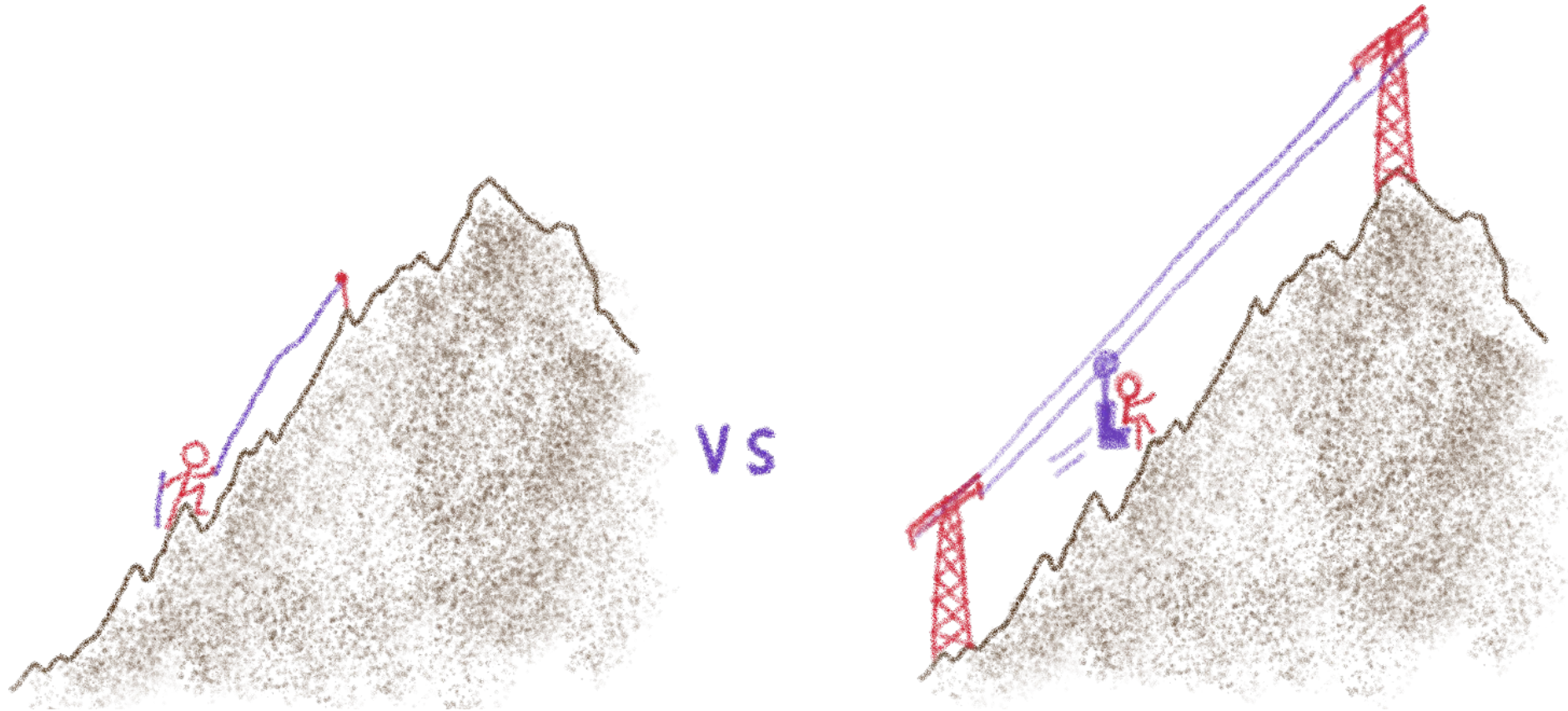
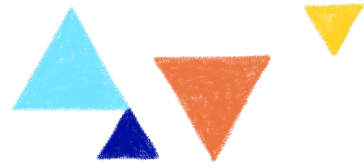
```
1 import { Component, Prop, h } from '@stencil/core';
2 import { format } from '../utils/utils';
3
4 @Component({
5   tag: 'my-component',
6   styleUrls: 'my-component.css',
7   shadow: true
8 })
9 export class MyComponent {
10   /**
11    * The first name
12    */
13   @Prop() first: string;
14
15   /**
16    * The middle name
17    */
18   @Prop() middle: string;
19
20   /**
21    * The last name
22    */
23   @Prop() last: string;
24
25   private getText(): string {
26     return format(this.first, this.middle, this.last);
27   }
28
29   render() {
30     return <div>Hello, World! I'm {this.getText()}</div>;
31   }
32 }
33
```



And they shouldn't need to be



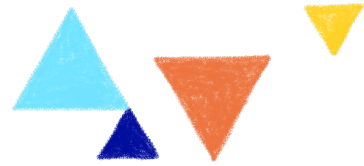
Go the extra mile to drive up adoption



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>

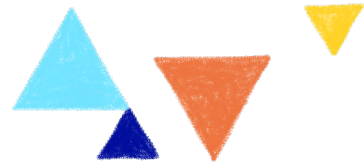
<lion-input name="firstName"></lion-input>
```



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();

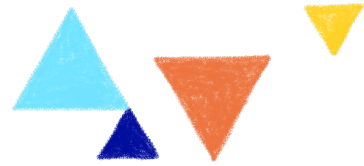
export const main = () => {
  return html` <lion-input-iban label="Account" name="account"></lion-input-iban> `;
};
```



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



Documentation isn't enough



The screenshot shows the Storybook website interface. On the left is a sidebar with a search bar and a navigation menu. The menu has sections for 'INTRO' and 'FORMS'. Under 'INTRO', 'Lion Web Components' is highlighted. Under 'FORMS', there is a list of components including 'Intro', 'Features Overview', 'Checkbox Group', 'Combobox', 'Fieldset', 'Field', 'Form', 'Input Amount', 'Input Date', 'Input Datpicker', 'Input Email', 'Input Iban', 'Input Range', and 'Input Stepper'. The main content area is titled 'Lion Web Components' and contains a paragraph describing the library, a link to an announcement blog post, a 'TODOs 124' badge, and a 'Demos' section with a link to a live Storybook. A 'Please note' section mentions Yarn Workspaces and the installation command 'yarn install'. The bottom of the main content area mentions 'Javascript tagged template literals' and 'lit-html engine'.

Storybook

Canvas Docs

Press "/" to search...

INTRO

- Lion Web Components**
- Announcement
- ⌵ Tabs Example

FORMS

- Intro
- ⌵ Features Overview
- ⌵ Checkbox Group
- ⌵ Combobox
- ⌵ Fieldset
- ⌵ Field
- ⌵ Form
- ⌵ Input Amount
- ⌵ Input Date
- ⌵ Input Datpicker
- ⌵ Input Email
- ⌵ Input Iban
- ⌵ Input Range
- ⌵ Input Stepper

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`.

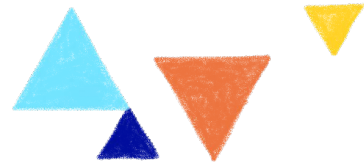
The code examples make use of [Javascript tagged template literals](#) which are a key component of the [lit-html engine](#) used in Lion. Additionally imports like `import '@lion/form/lion-form.js'` need to be transformed somehow, for example by [es-dev-server](#).



make adoption easy



Keeping a coherent writing style



Web Components guidelines at Clever Cloud

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

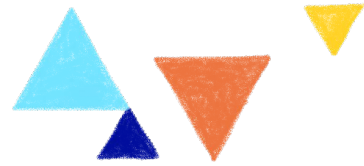
General rules and reminders

- 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



clever cloud

Press "/" to search...

HOME

- Readme
- Changelog
- Contributing
- Release

DOCS

- Architecture Decision Records
- How to translate and localize?**
- How to write stories
- Web components guidelines

ATOMS

- <cc-beta>
 - Default
 - With tiles
 - With map
- <cc-button>
- <cc-datetime-relative>
- <cc-expand>

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





Thank you all!

