

JFOKUS

Brown Bag Lunch

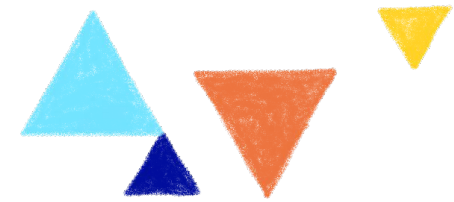
**Design, develop and manage
a catalog of Web Components**

Horacio Gonzalez

2021-03-23

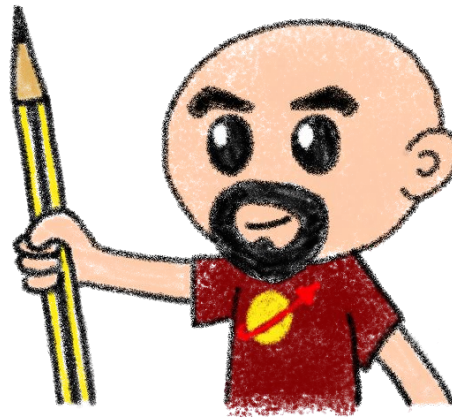


@LostInBrittany



Who are we?

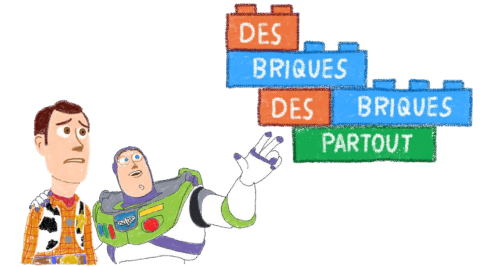
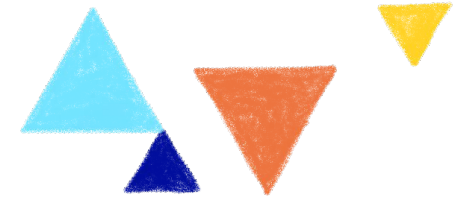
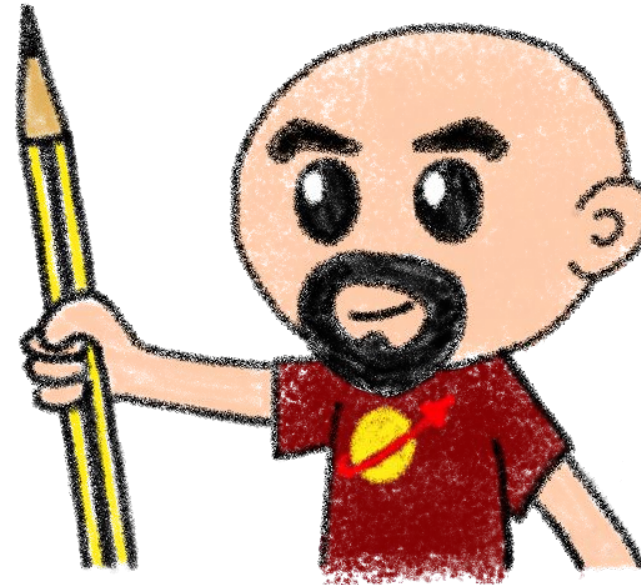
Introducing myself and
introducing OVHcloud



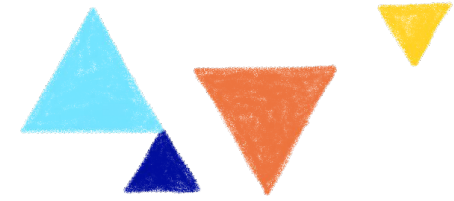
Horacio Gonzalez

@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



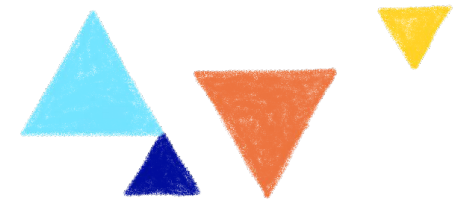
1.5 Billion Euros Invested
since 2016



P.U.E. 1.09
Energy efficiency indicator



20 Years in Business
Disrupting since 1999

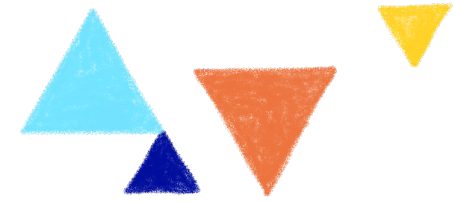


The 3 minutes context

What the heck are web component?

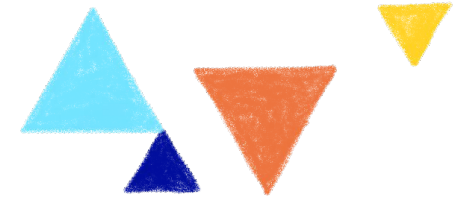


Web Components



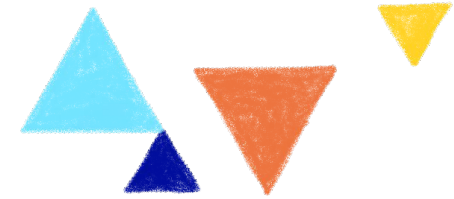
Web standard W3C

Web Components



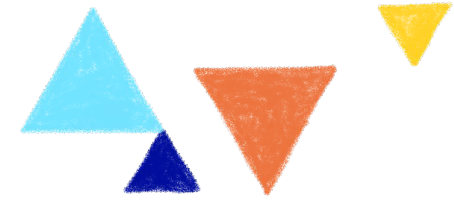
Available in all modern browsers:
Firefox, Safari, Chrome

Web Components



Create your own HTML tags
Encapsulating look and behavior

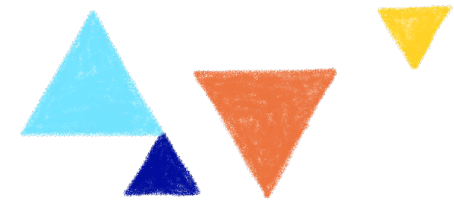
Web Components



Fully interoperable

With other web components, with any framework

Web Components



CUSTOM ELEMENTS

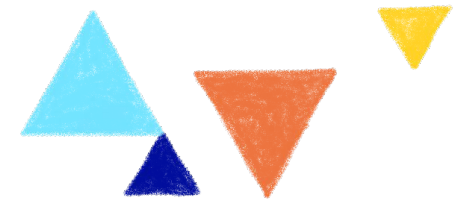


SHADOW DOM



TEMPLATES

Custom Element



To define your own HTML tag

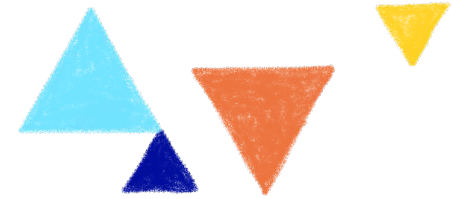
```
<body>
  ...
  <script>
    window.customElements.define('my-element',
      class extends HTMLElement {...});
  </script>
  <my-element></my-element>
</body>
```

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>
```

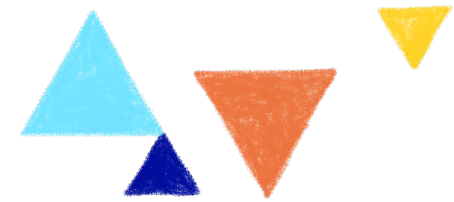


Hello, world!



こんにちは、影の世界!

Template

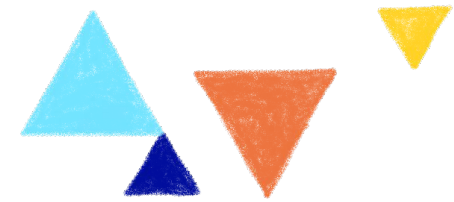


To have clonable document template

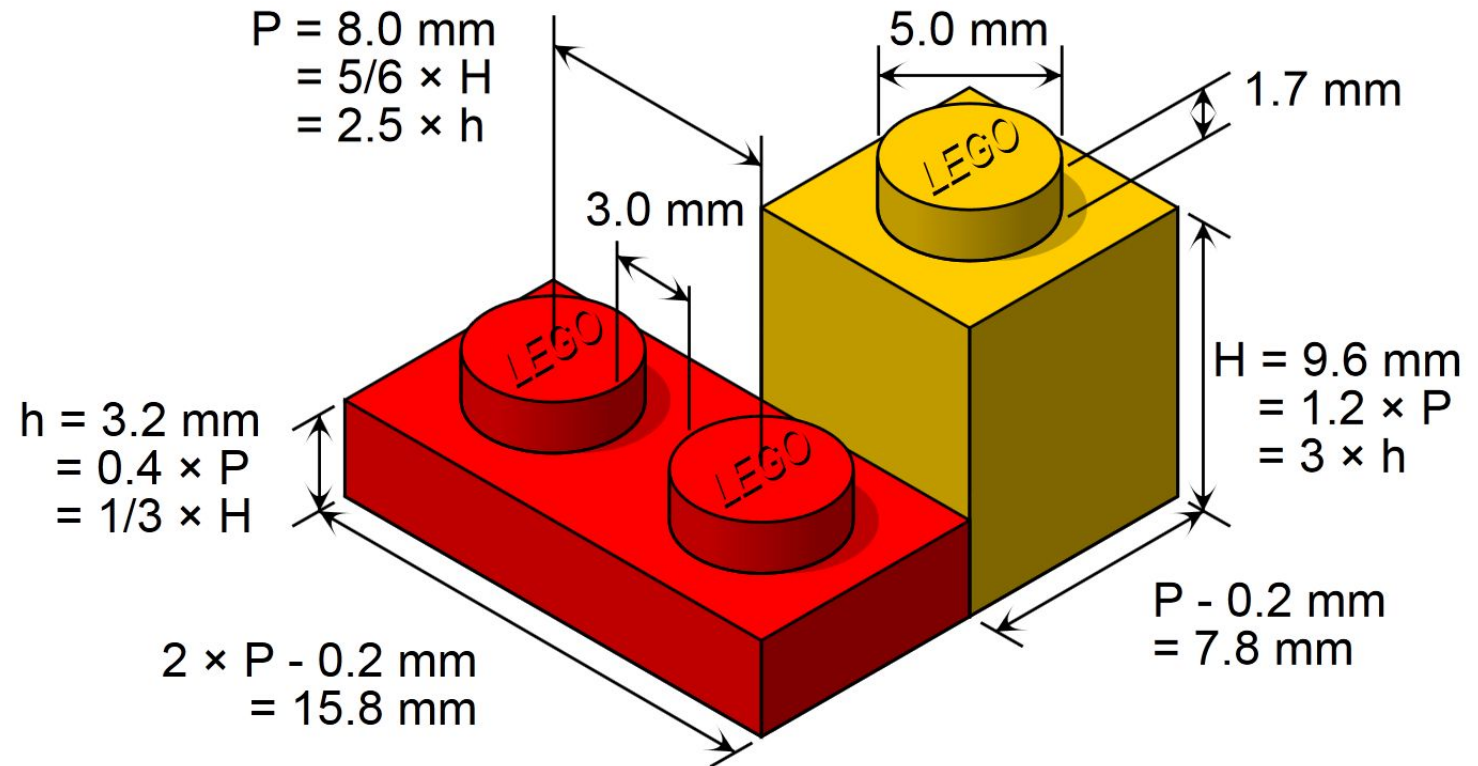
```
<template id="mytemplate">  
  <img src="" alt="great image">  
  <div class="comment"></div>  
</template>
```

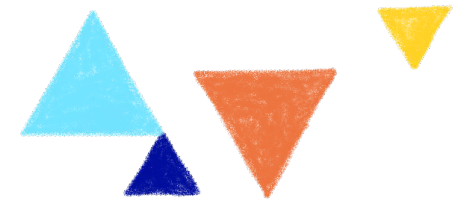
```
var t = document.querySelector('#mytemplate');  
// Populate the src at runtime.  
t.content.querySelector('img').src = 'logo.png';  
var clone = document.importNode(t.content, true);  
document.body.appendChild(clone);
```

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.

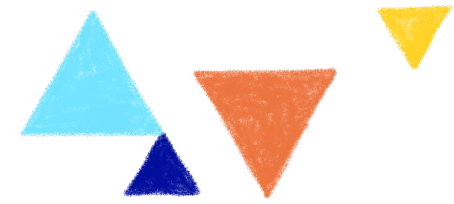
Expressive

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

Color theme creation

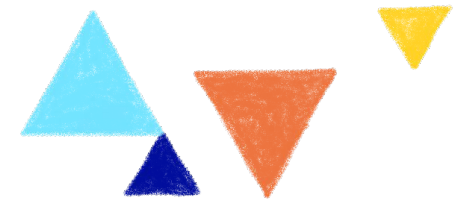
- Color usage and palettes
- Color theme creation**
- Tools for picking colors

A talk for devs by a dev



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

The same or different?



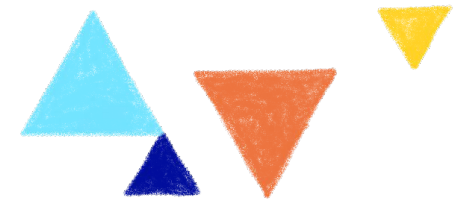
Design System

Component Catalog

Style Guide

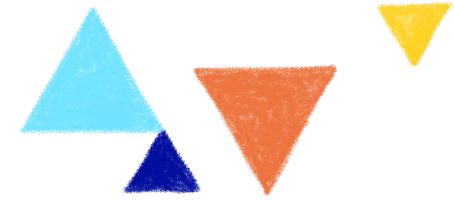


Style Guides



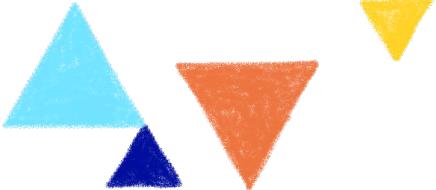
A **document** listing the **styles**, **patterns**, **practices**, and **principles** of a brand **design standards**

Style Guides



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

Style Guide Example: Uber



Uber

Brand story System elements Showcase Downloads Help / FAQ

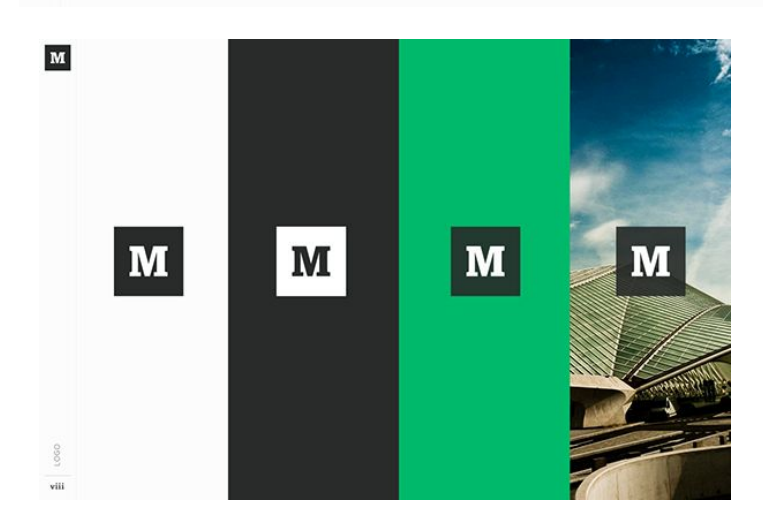
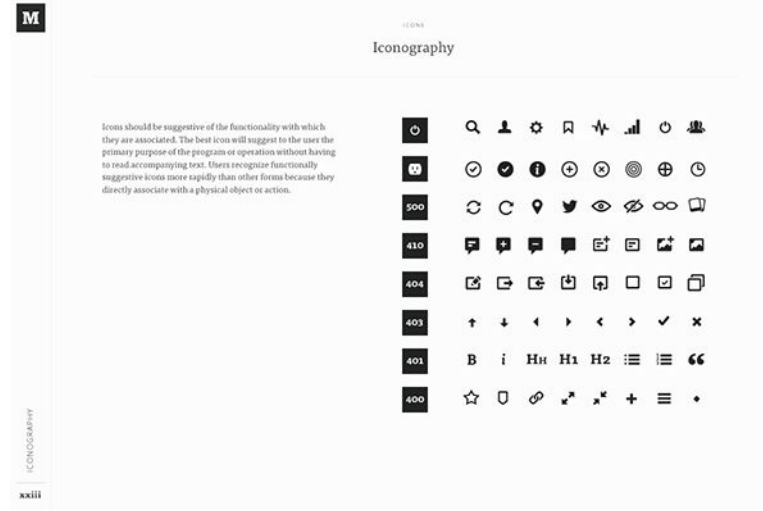
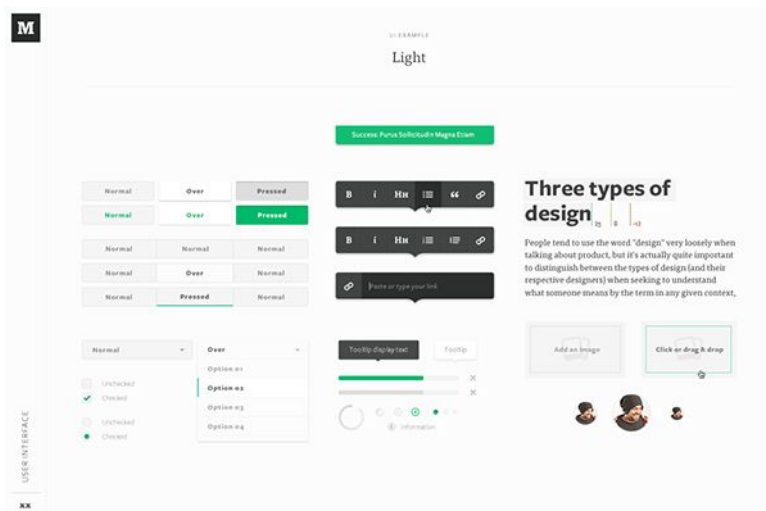
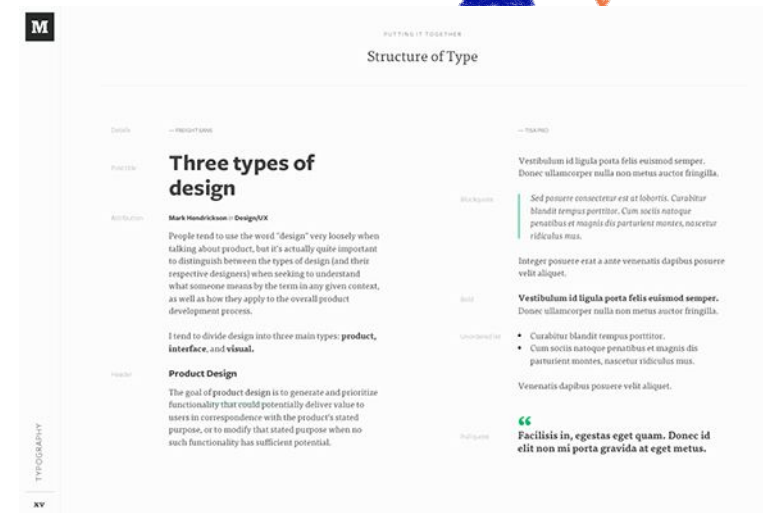
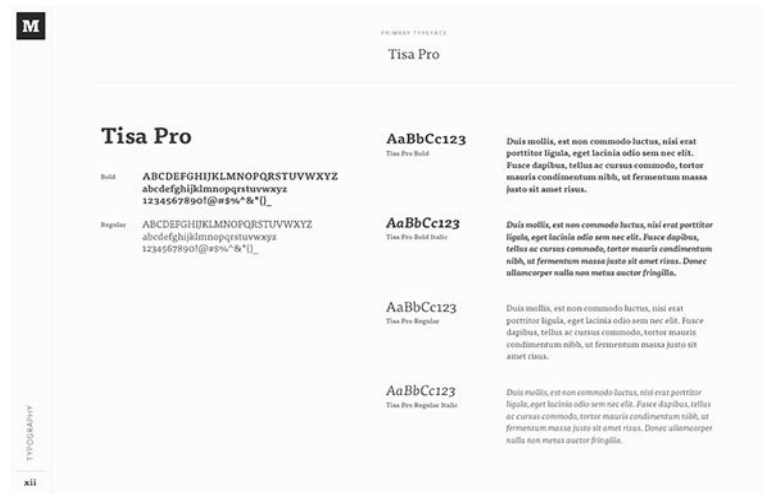
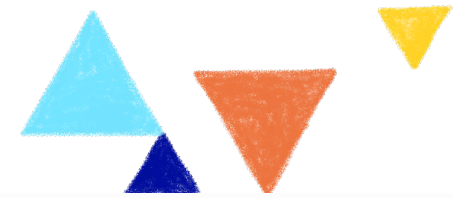
Primary Lockup

Uber Primary

Display Medium Display Medium

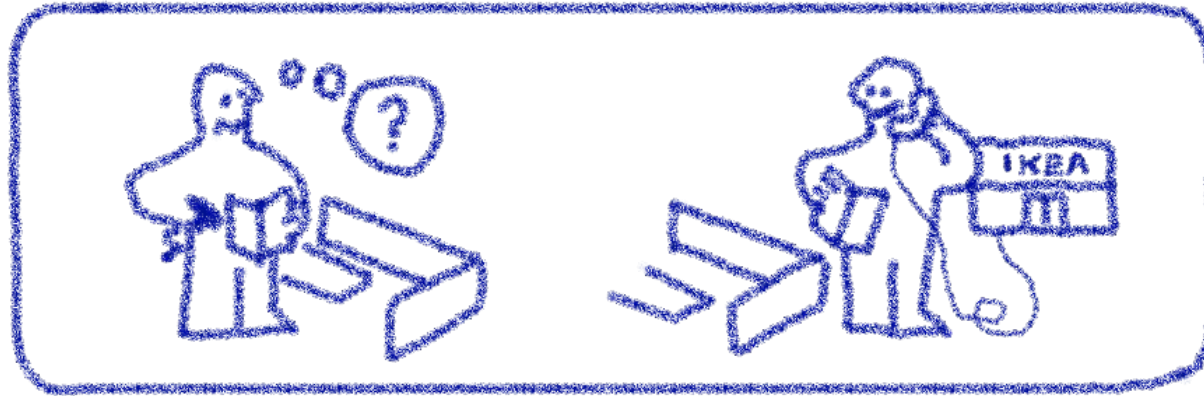
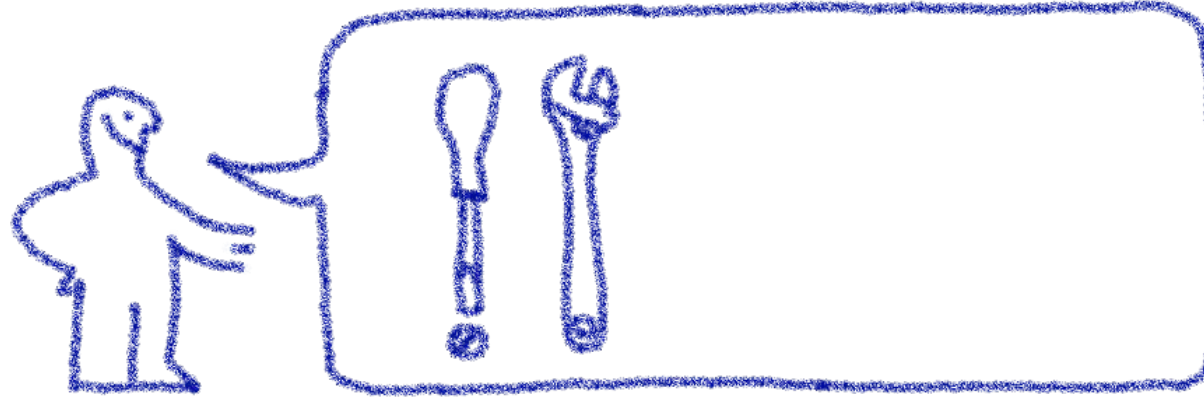
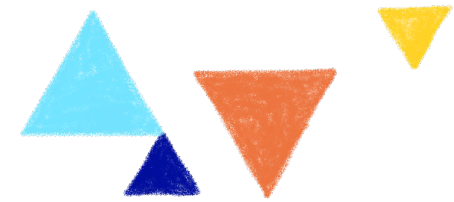
<https://brand.uber.com/>

Style Guide Example: Medium



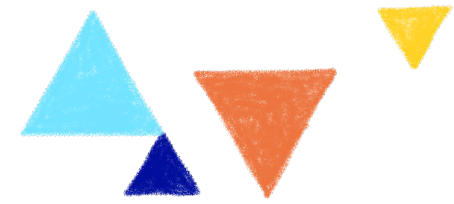
<https://www.behance.net/gallery/7226653/Medium-Brand-Development>

Style Guides alone are ambiguous



Interpretation needed to adapt the preconisation to the use case

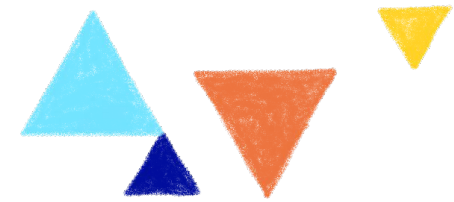
Component Catalogs



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!

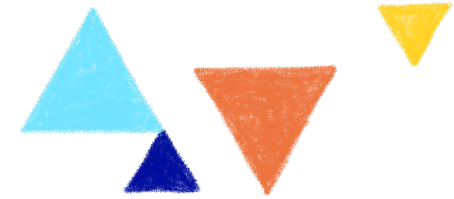
A simple dark alert—check it out!

```
<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>
```



<https://getbootstrap.com/>

Component Catalog Example: ING's Lion



Storybook Canvas Docs

Q Press "/" to search...

INTRO

- Lion Web Components
- Announcement
- Tabs Example

FORMS

- Intro
- Features Overview
- Checkbox Group
- Fieldset
- Field
- Form
- Input Amount
- Input Date
- ▾ Input Datepicker
 - Main**
 - Minimum And Maximum Date
 - Disable Specific Dates
 - Calendar Heading
 - Disabled
 - Read Only

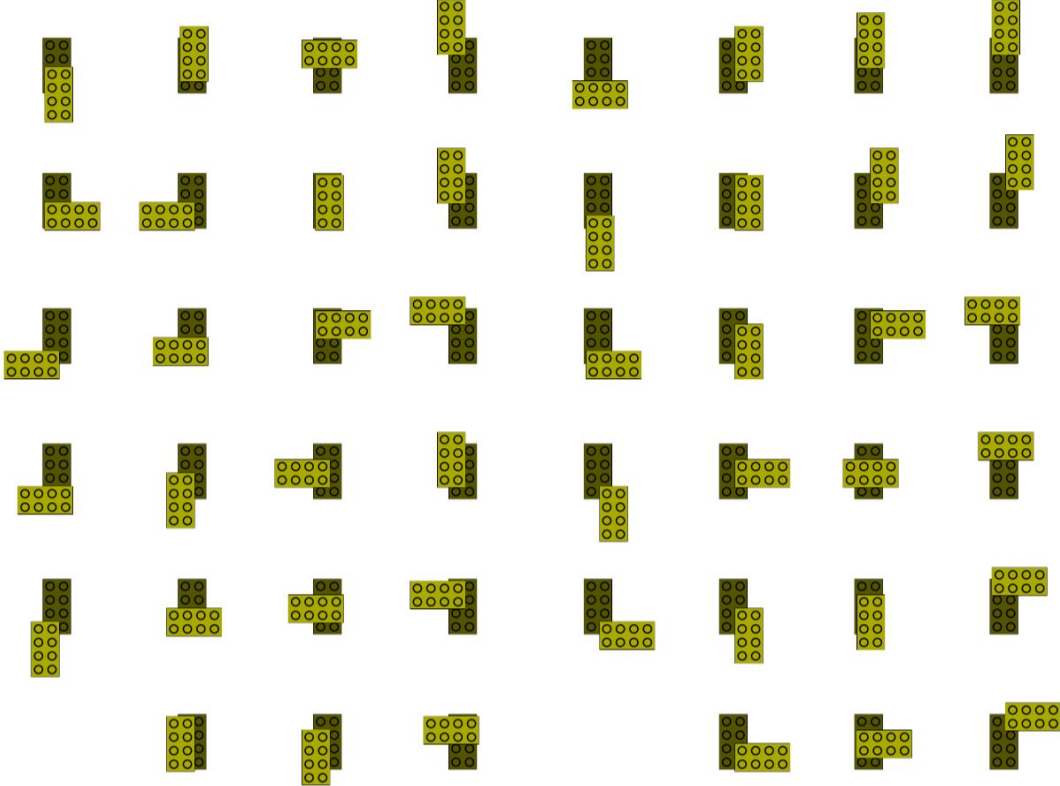
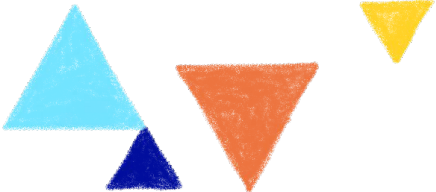
Date

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

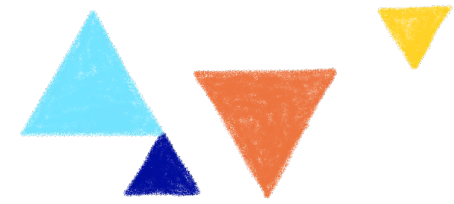
<https://lion-web-components.netlify.app/>

Catalogs alone create inconsistency



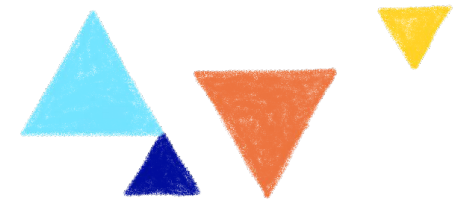
Like using the same LEGO bricks to create very different objects

Design Systems



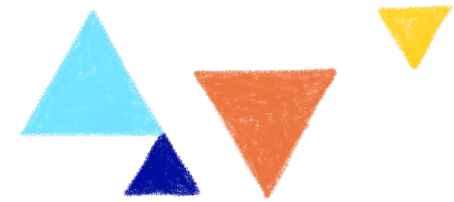
A Design System is like a **common visual language** for **product teams**

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

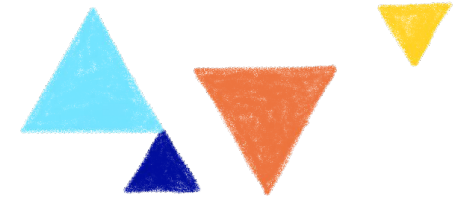


Design System \approx

Style Guide + Component Catalog



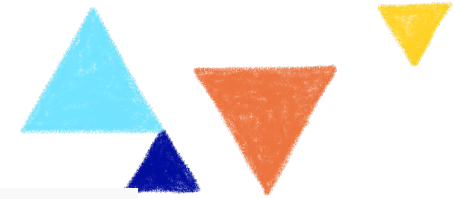
Example: Carbon Design System



The screenshot shows the Carbon Design System website. The top navigation bar is dark with the text 'Carbon Design System' on the left, a search icon, and a grid icon on the right. A sidebar on the left contains a list of navigation items: 'Get started' (with an up arrow), 'About Carbon' (highlighted with a blue bar), 'Design', 'Develop', 'Tutorial' (with a down arrow), 'Guidelines' (with a down arrow), 'Components' (with a down arrow), 'Patterns' (with a down arrow), 'Data visualization' (with a down arrow), 'Resources', 'How to contribute' (with a down arrow), 'Updates' (with a down arrow), 'Help' (with a down arrow), and 'Community' (with a down arrow). Below these items are 'Design kit' and 'GitHub' (with an external link icon). The main content area has a dark header with the text 'About Carbon' in white. Below this, the text reads: '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.' At the bottom of the main content area, there is a list of links: '↳ Introduction', '↳ Guiding principles', '↳ Governance', and '↳ 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

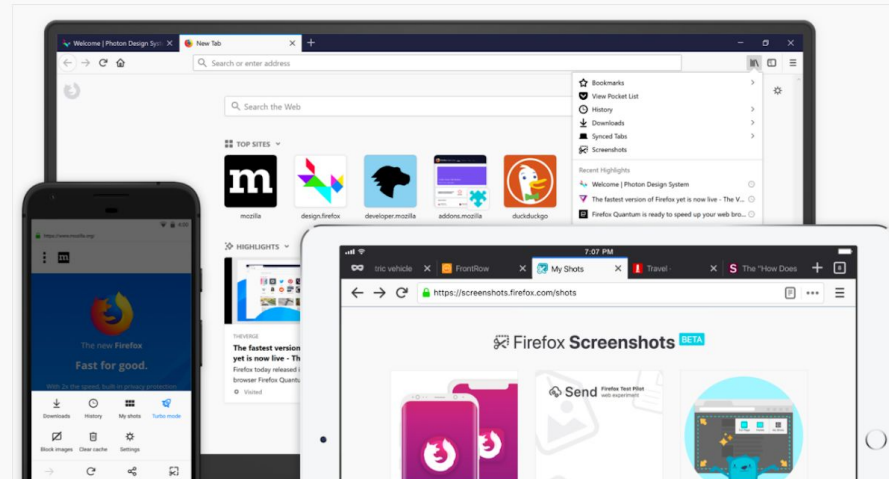
Patterns

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.

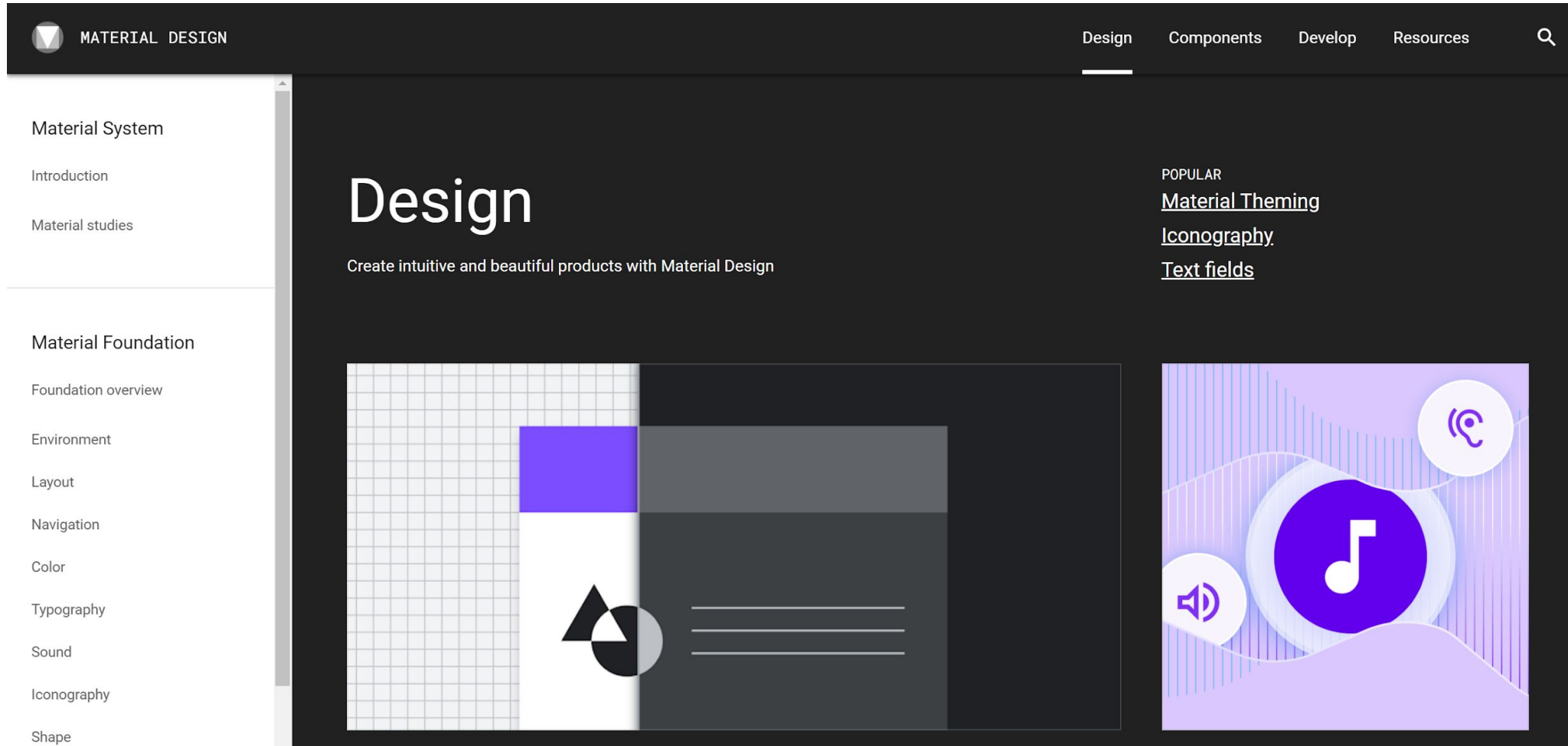
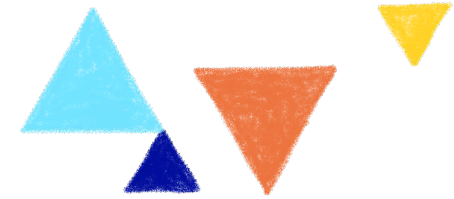
The Photon Design System houses guidelines, reusable UI components, templates, and other resources to help you create products for Firefox users. It is flexible and always evolving to serve the best Firefox experience for every situation.

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

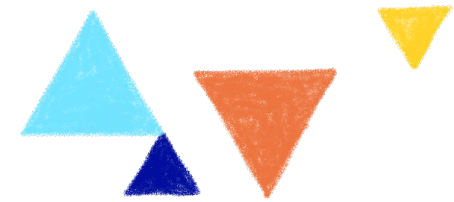
You can help us improve the system and ensure it remains current and relevant.

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

Example: Material Design



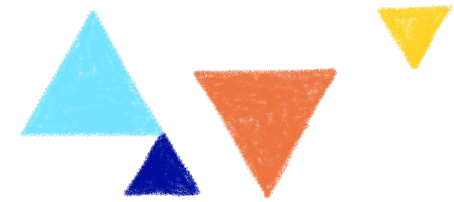
<https://material.io/>



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!

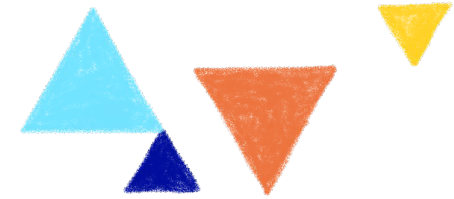
A simple dark alert—check it out!

```
<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







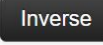

A long time ago



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.

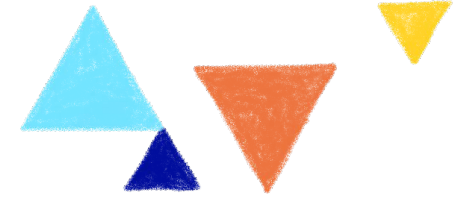
Button	class=""	Description
	<code>btn</code>	Standard gray button with gradient
	<code>btn btn-primary</code>	Provides extra visual weight and identifies the primary action in a set of buttons
	<code>btn btn-info</code>	Used as an alternative to the default styles
	<code>btn btn-success</code>	Indicates 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](#)

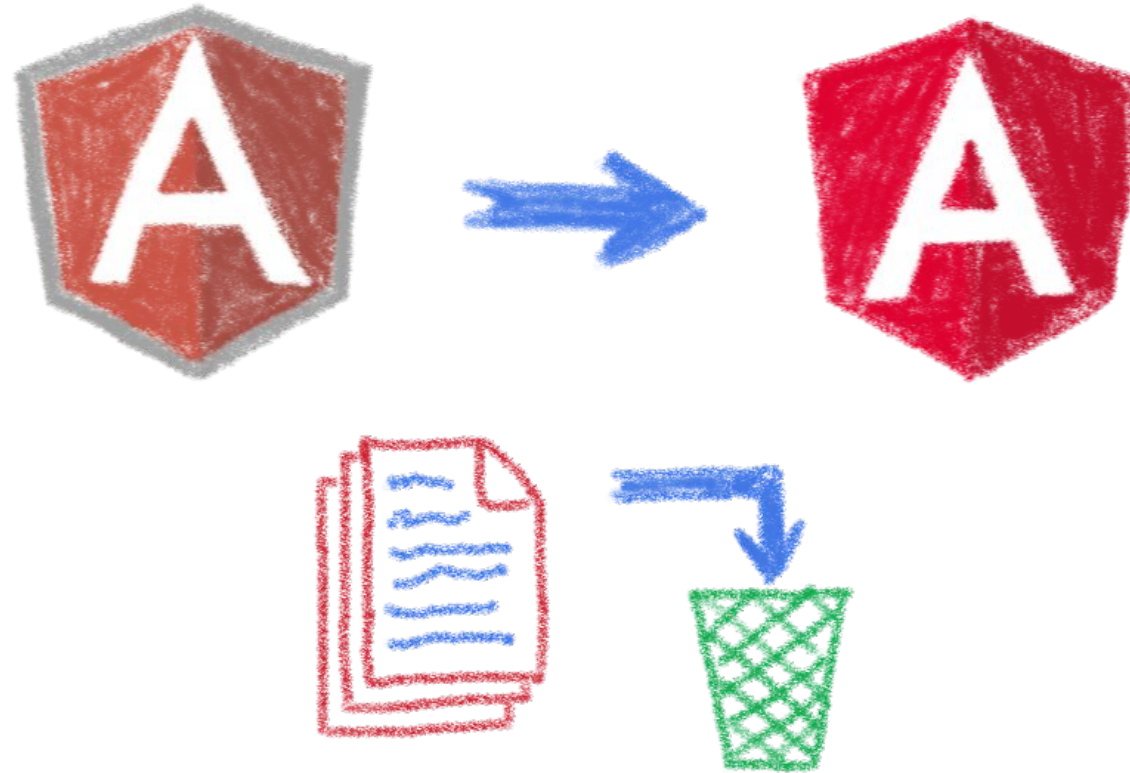
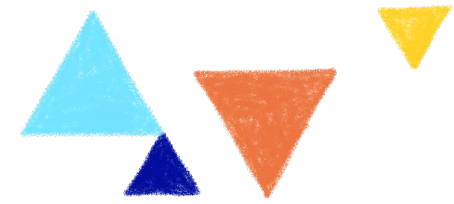
[Star 14,640](#) [Fork 7,184](#) [Tweet](#)



Getting started

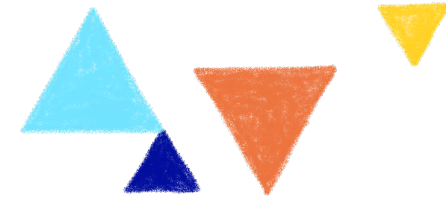
And new reference implementations were needed

But you know the sad story...



All UI Bootstrap based catalogs woke up with an obsolete implementation

Worry no more, let's do Angular!



Home Getting Started Components

1240914
npm



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](#)

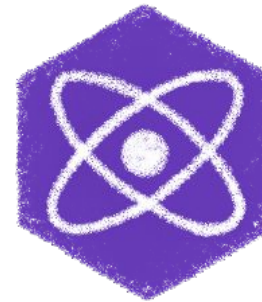
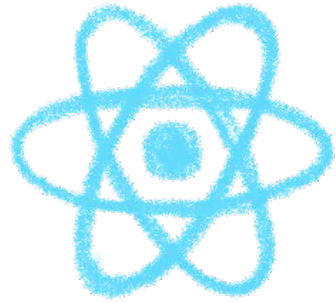
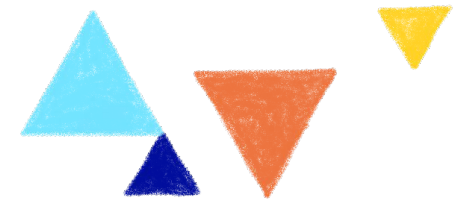
Currently at v6.1.0



ng-bootstrap to the rescue



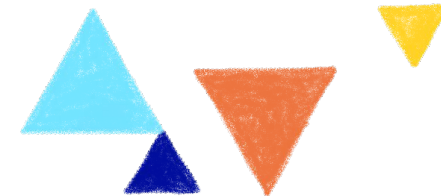
But times had changed...



In 2017 Angular is only one more in the clique



React is the new Big Thing™



Black Lives Matter. [Support the Equal Justice Initiative.](#)


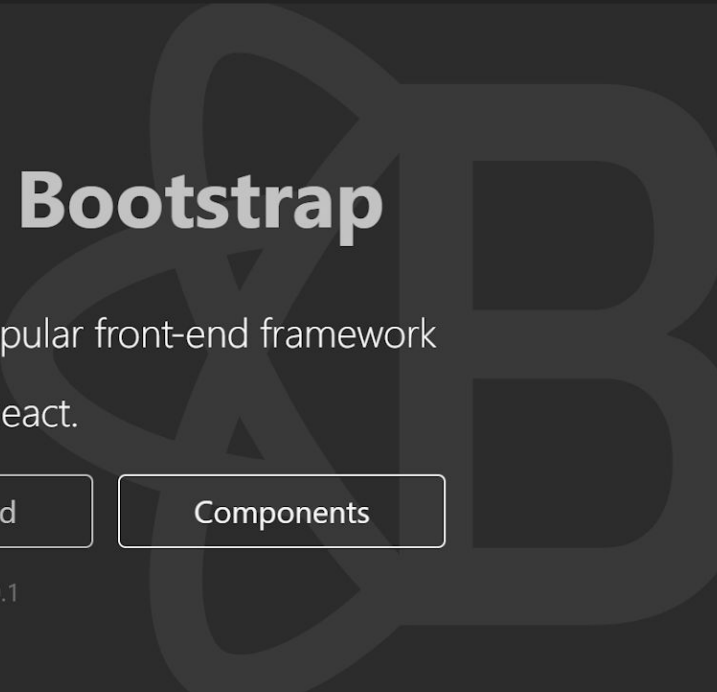
Home Getting Started Components v1.0.1 (Bootstrap 4.5)

React Bootstrap

The most popular front-end framework
Rebuilt for React.

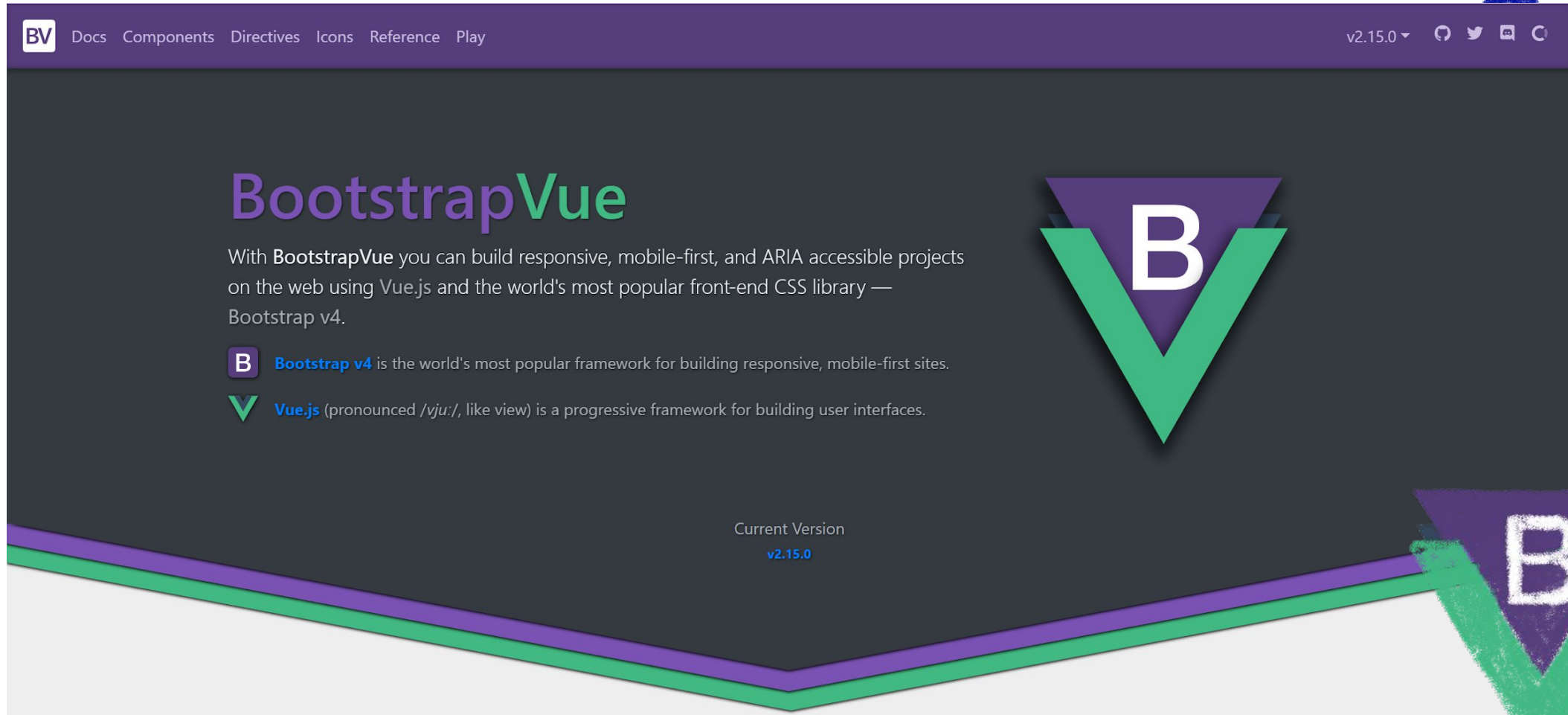
[Get started](#) [Components](#)

Current version: 1.0.1



So let's build React Bootstrap...

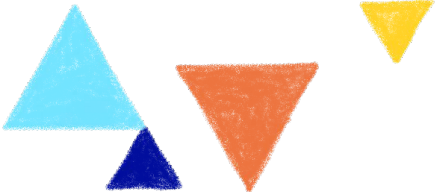
Wait, what about ue?



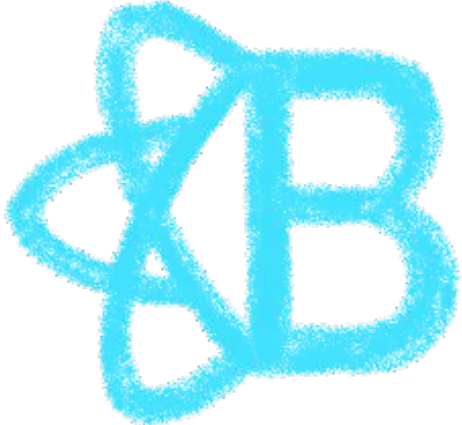
The screenshot shows the BootstrapVue website interface. At the top, there is a navigation bar with the 'BV' logo and links for 'Docs', 'Components', 'Directives', 'Icons', 'Reference', and 'Play'. The version 'v2.15.0' is displayed in the top right corner along with social media icons. The main content area features the 'BootstrapVue' title in purple and green. Below the title, a paragraph explains that BootstrapVue allows building responsive, mobile-first, and ARIA accessible projects using Vue.js and Bootstrap v4. Two callout boxes are present: one with a purple 'B' icon for Bootstrap v4 and another with a green 'V' icon for Vue.js. At the bottom of the main content, it states 'Current Version v2.15.0'. The entire screenshot is overlaid with a large, stylized 'BV' logo in purple and green.

We also need BootstrapVue

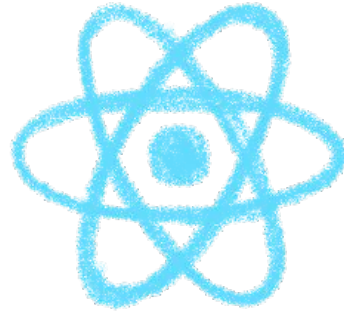
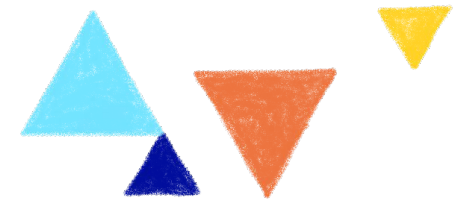
OK, I think you see my point...



Bootstrap

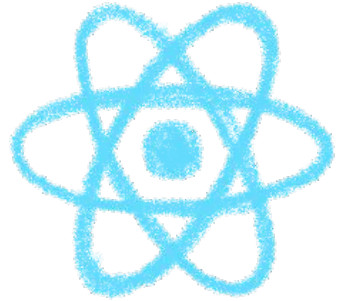
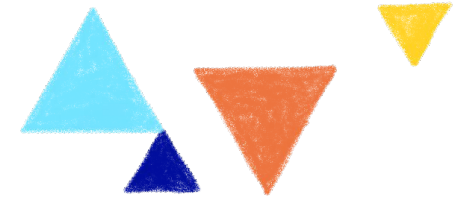


Most Design System do a choice



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

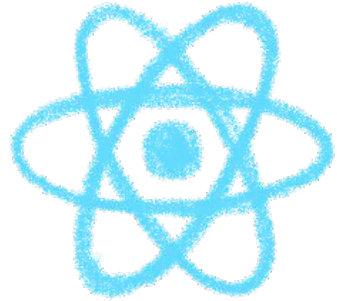
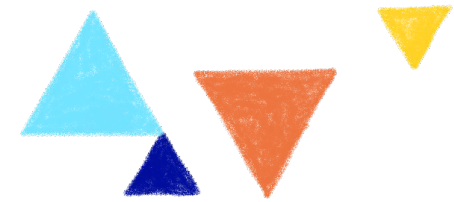
Both choices are problematic



Shipping only one implementation:

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

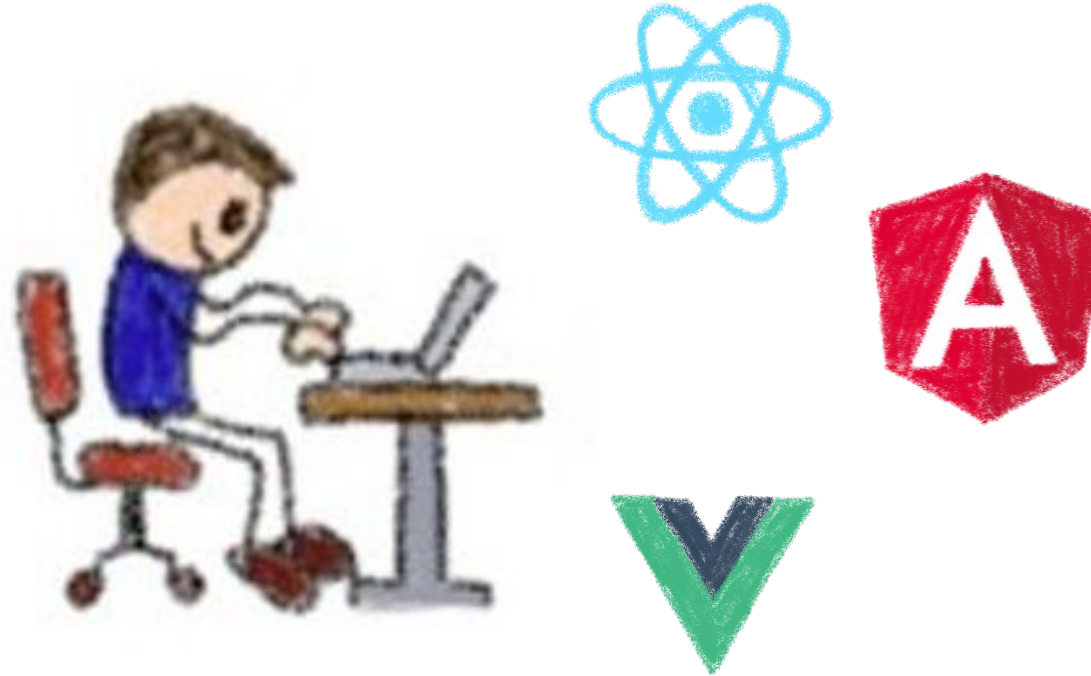
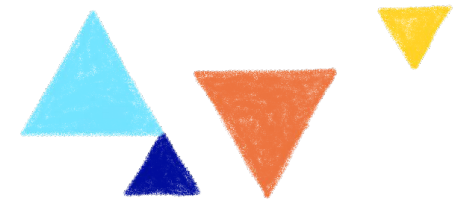
Both choices are problematic



Shipping several implementations:

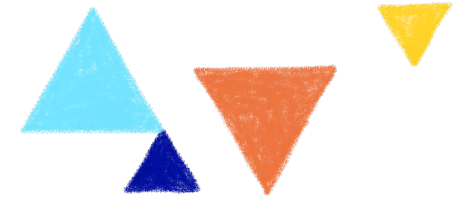
You need to maintain all the implementation...
and you still miss some others

Incomplete catalogs are problematic



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

Example: Carbon Design System



The screenshot shows the Carbon Design System website. The header is dark with the text 'Carbon Design System' and search and menu icons. A left sidebar contains a navigation menu with items like 'Get started', 'About Carbon', 'Design', 'Develop', 'Tutorial', 'Guidelines', 'Components', 'Patterns', 'Data visualization', 'Resources', 'How to contribute', 'Updates', 'Help', 'Community', 'Design kit', and 'GitHub'. The main content area has a large black header with 'About Carbon' in white. Below this is a paragraph of text and a list of links. To the right of the text are three large, stylized icons: a blue atom, a red shield with a white 'A', and a green 'V'. The bottom right corner of the page features a decorative graphic of colorful triangles.

Carbon Design System

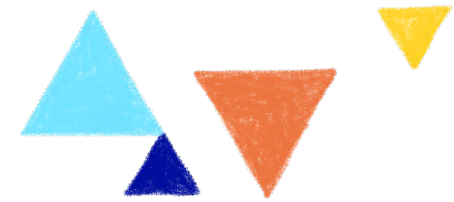
- Get started
- About Carbon
- Design
- Develop
- Tutorial
- Guidelines
- Components
- Patterns
- Data visualization
- Resources
- How to contribute
- Updates
- Help
- Community

- 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

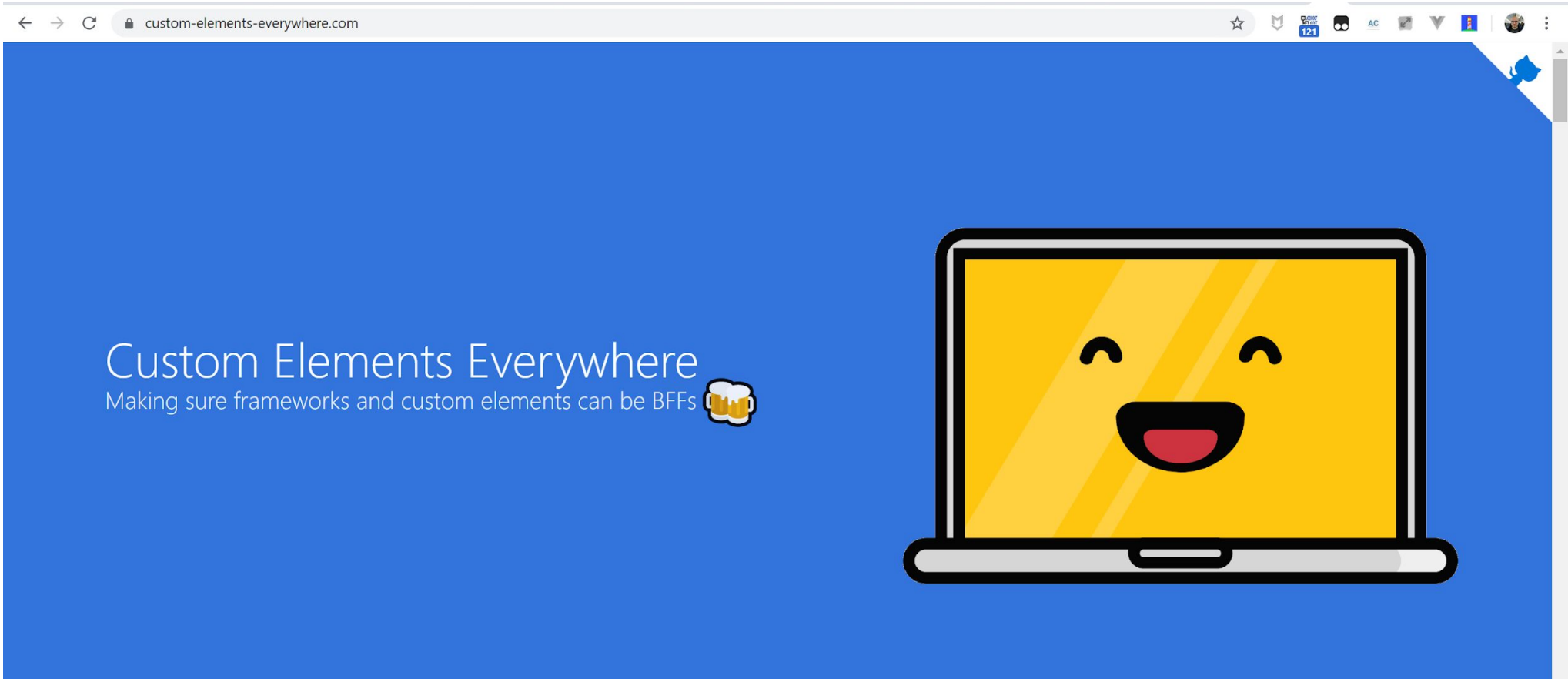
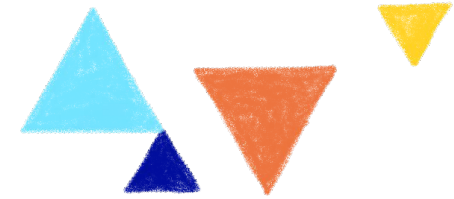


Web Components & Design Systems

A match made in heaven

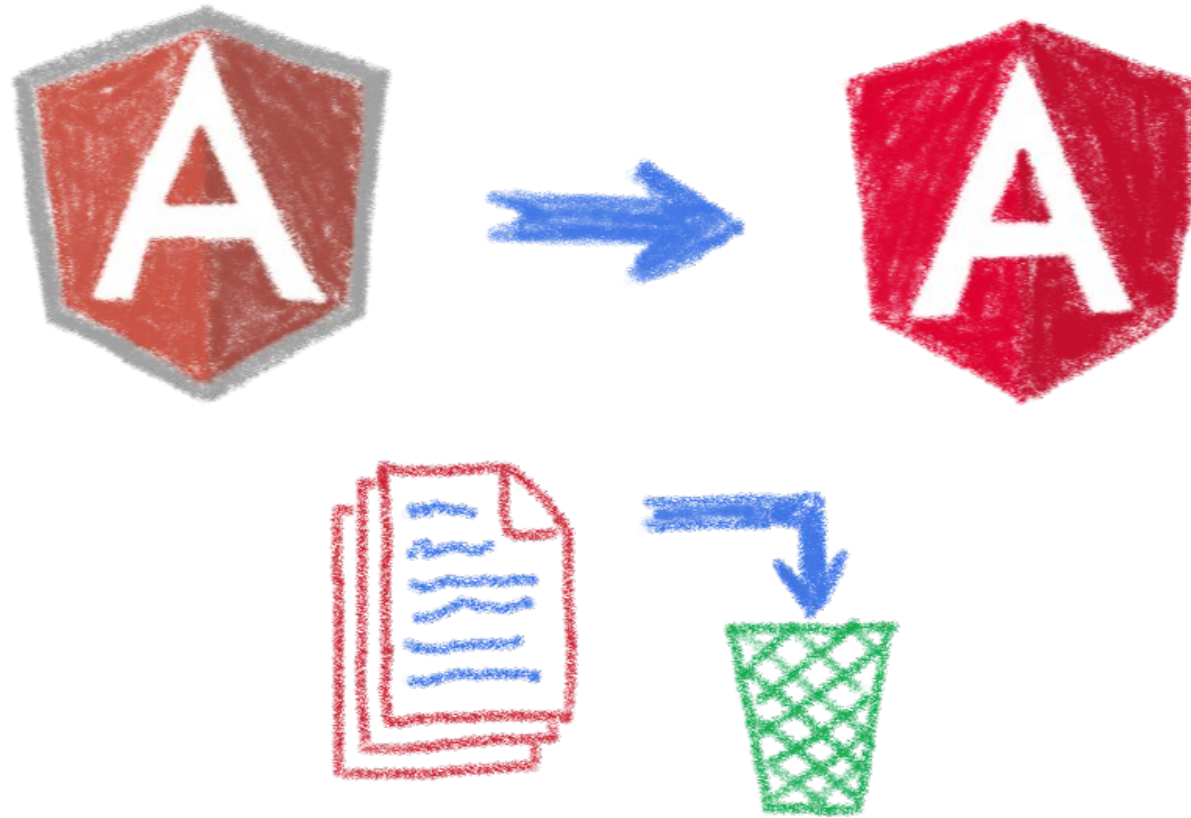
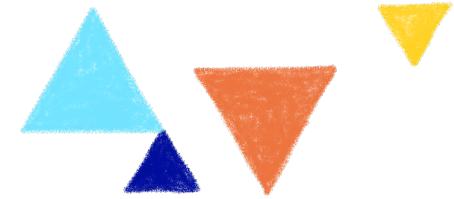


Compatibility is on Web Components side



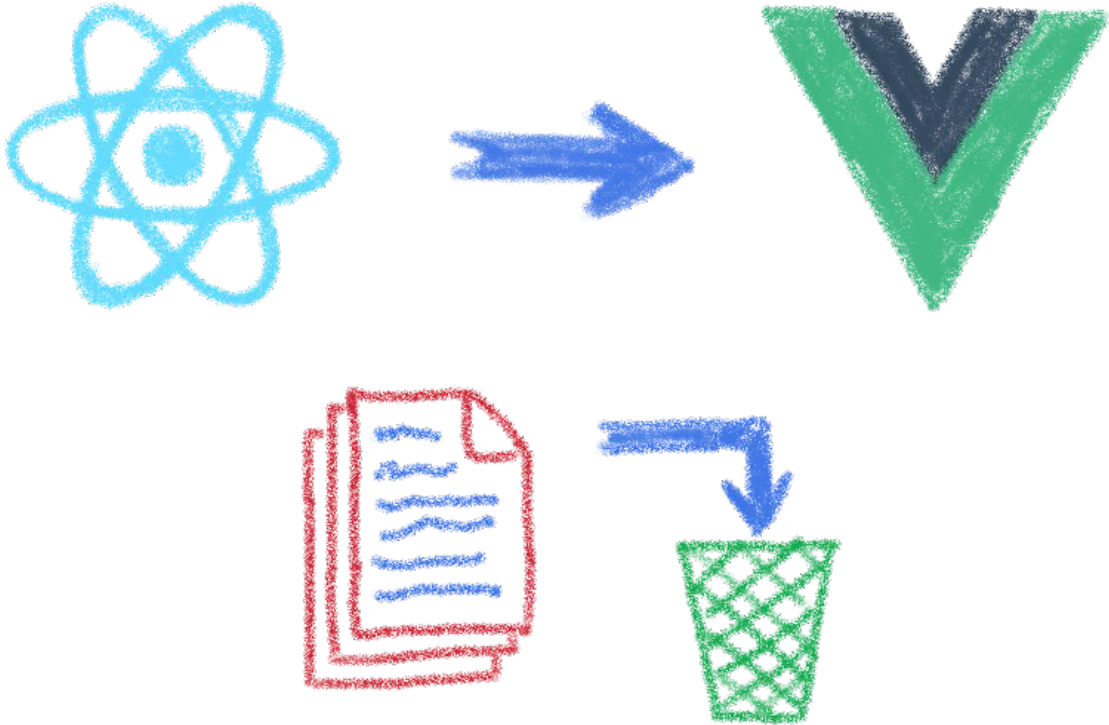
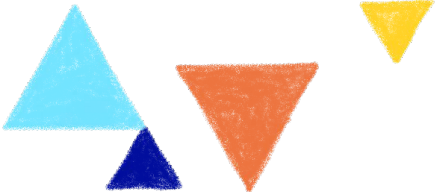
Web Components everywhere, baby!

Do you remember AngularJS?



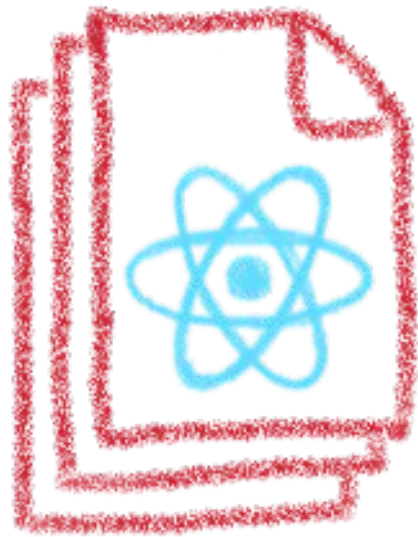
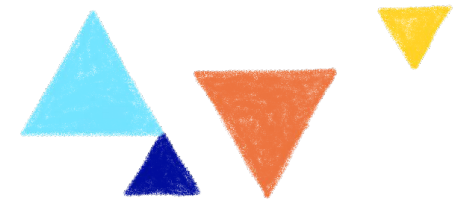
And all the code put in the trash bin
when Angular arrived...

The pain of switching frameworks?



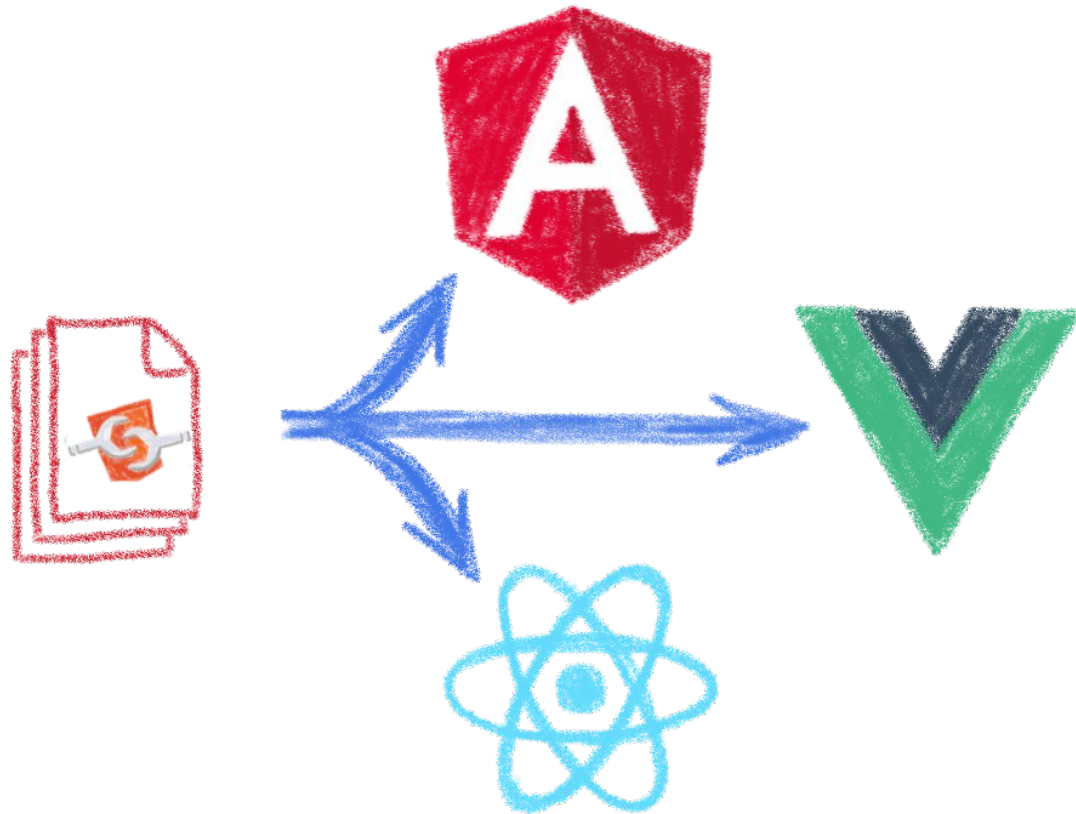
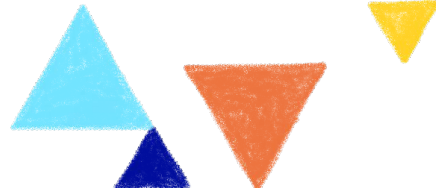
Rewriting once again your code...

The impossibility of sharing UI code?



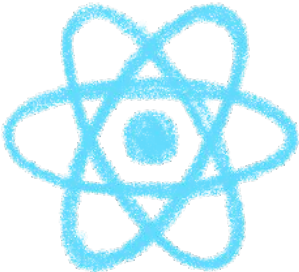
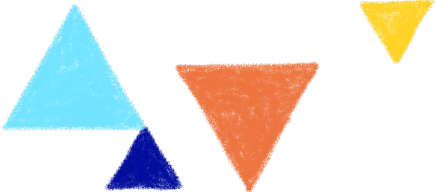
Between apps written with different frameworks

Web Components change that



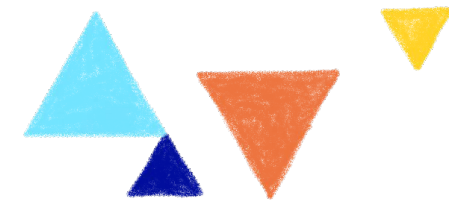
In a clean and standard way

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

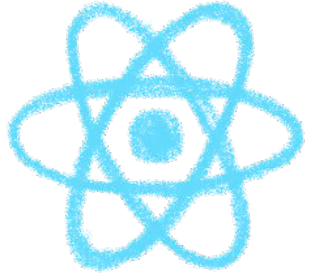
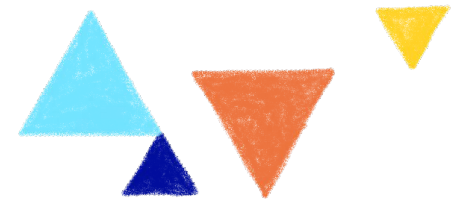
We use a reactive programming library that we developed in house. - Sofian

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

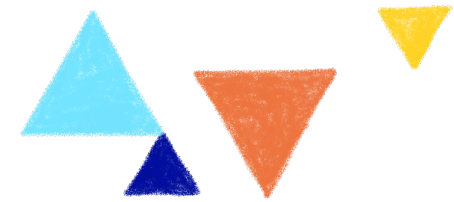
 Even in the spaaaaaaace 

You can have a single implementation

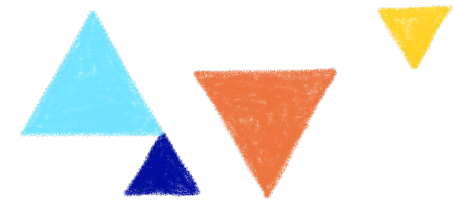


And it simply works everywhere

When you need interoperability



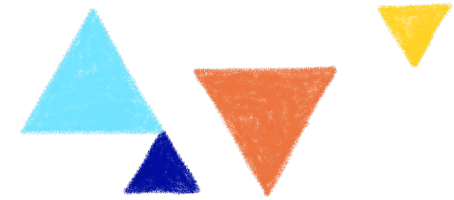
Nothing beats the standard



But how to do it?

Designing, developing and managing
a catalog of Web Components

Learning from the best

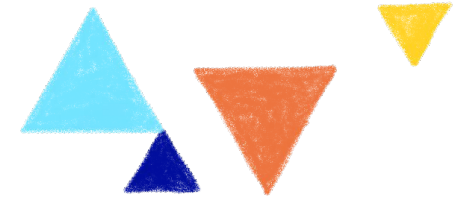


The screenshot shows the Storybook interface for a 'Date' component. The component is displayed as a calendar for June 2020. The date 22 is highlighted with a green border. The left sidebar contains a navigation menu with the following items:

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

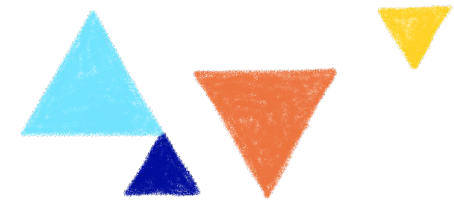
<https://lion-web-components.netlify.app/>

Learning from the best



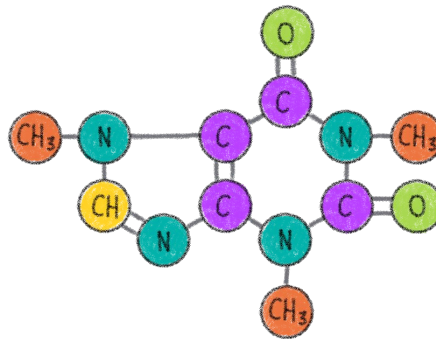
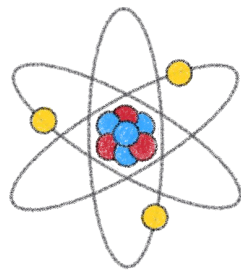
The screenshot shows the Clever Cloud documentation interface. On the left is a sidebar with a search bar and navigation menus for 'HOME', 'DOCS', and 'ATOMS'. The 'HOME' menu includes 'Readme', 'Changelog', 'Contributing', and 'Release'. The 'DOCS' menu includes 'Architecture Decision Records', 'How to translate and localize?', 'How to write stories', and 'Web components guidelines'. The 'ATOMS' menu lists various web components like '<cc-beta>', '<cc-button>', '<cc-datetime-relative>', '<cc-expand>', '<cc-flex-gap>', and '<cc-img>'. The main content area displays the title 'Collection of Web Components by Clever Cloud' and sections for 'What is this?' and 'Why is it public?'. The 'What is this?' section explains that the project contains a collection of Web Components made by Clever Cloud, some low-level like '<cc-button>', '<cc-input-text>', or '<cc-loader>', and others high-level and specific to Clever Cloud's domain model. The 'Why is it public?' section lists three reasons: to share knowledge and experience, to allow clients and partners to use them in their own custom Web UIs, and to provide a feedback mechanism.

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

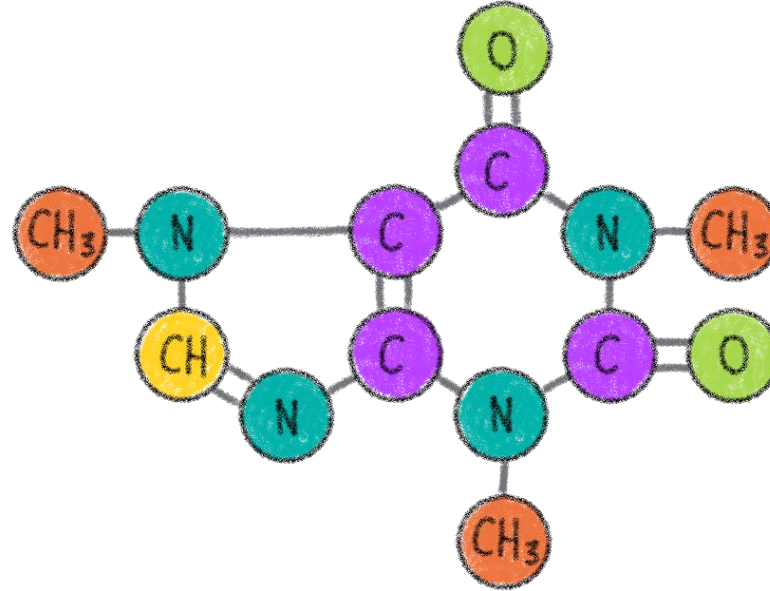
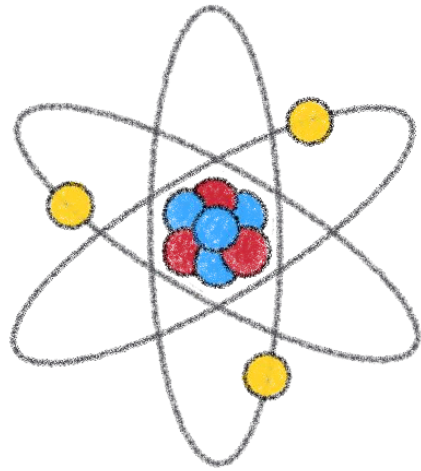
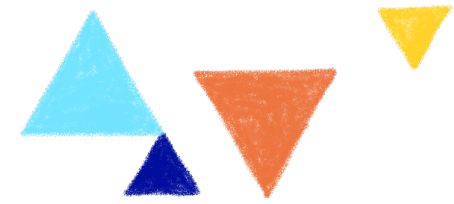


What kind of components?

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

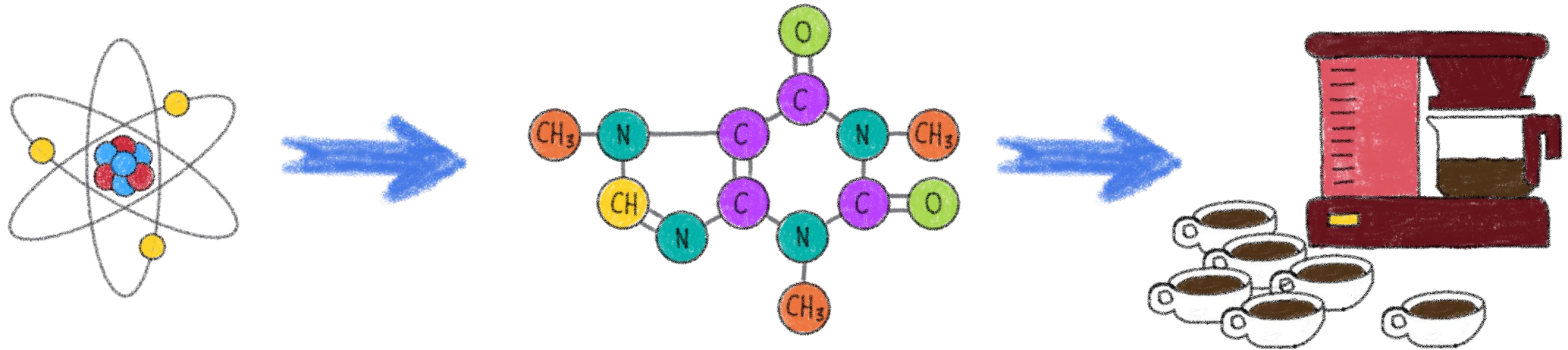
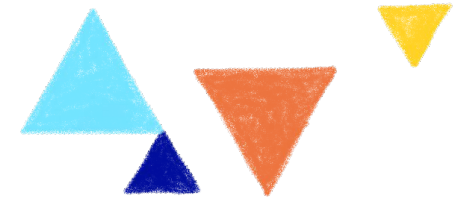


A matter of size and complexity



What kind(s) of components you want to build

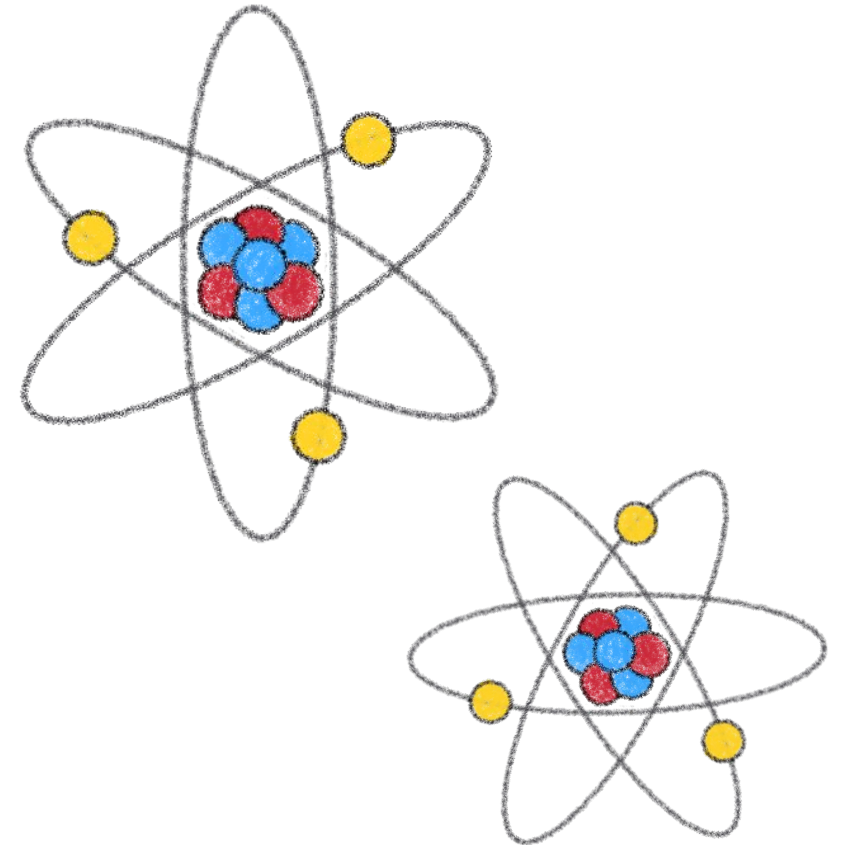
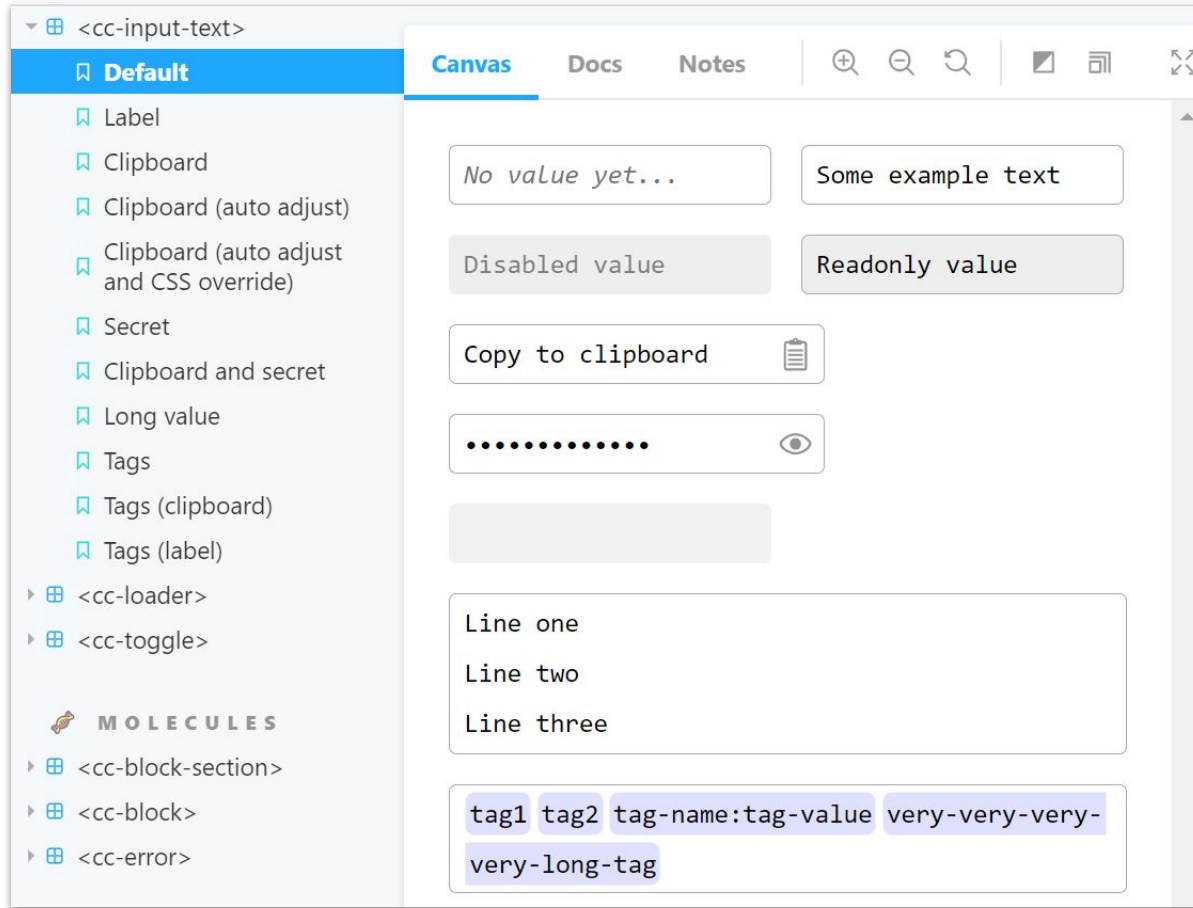
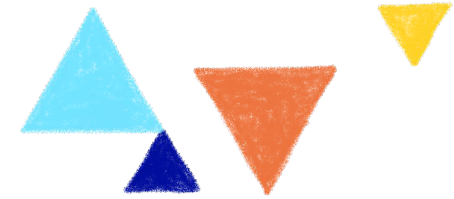
Build from the bottom and go up



Eat your own dog food

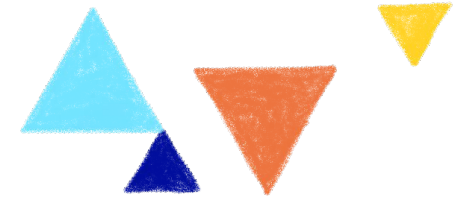


And how to choose the atoms?



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

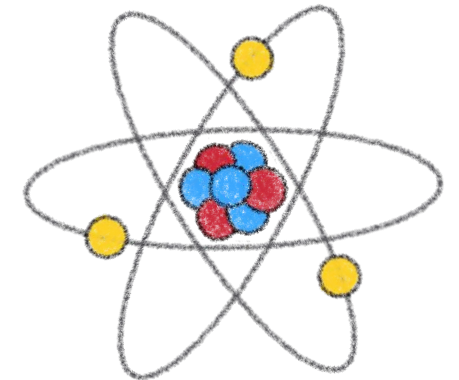
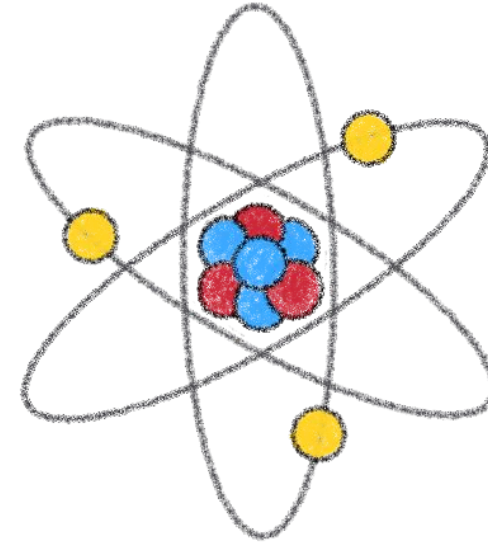
`lion-input-iban` component is based on the generic text input field. Its purpose is to provide a way for users to fill in an IBAN (International Bank Account Number).

Account

Show code

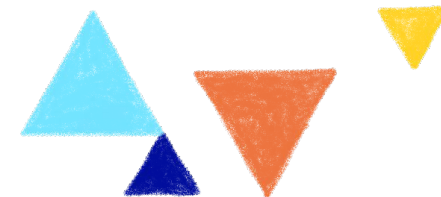
Features

- 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?



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

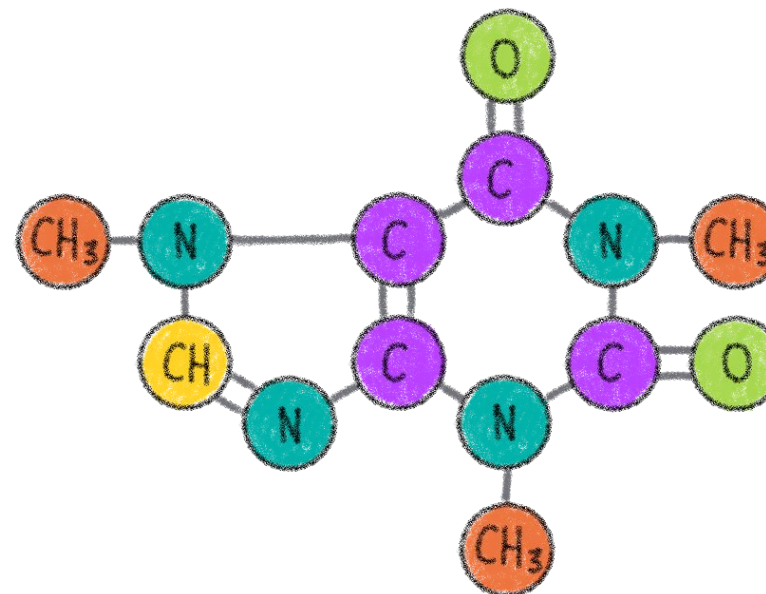
MOLECULES

> <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

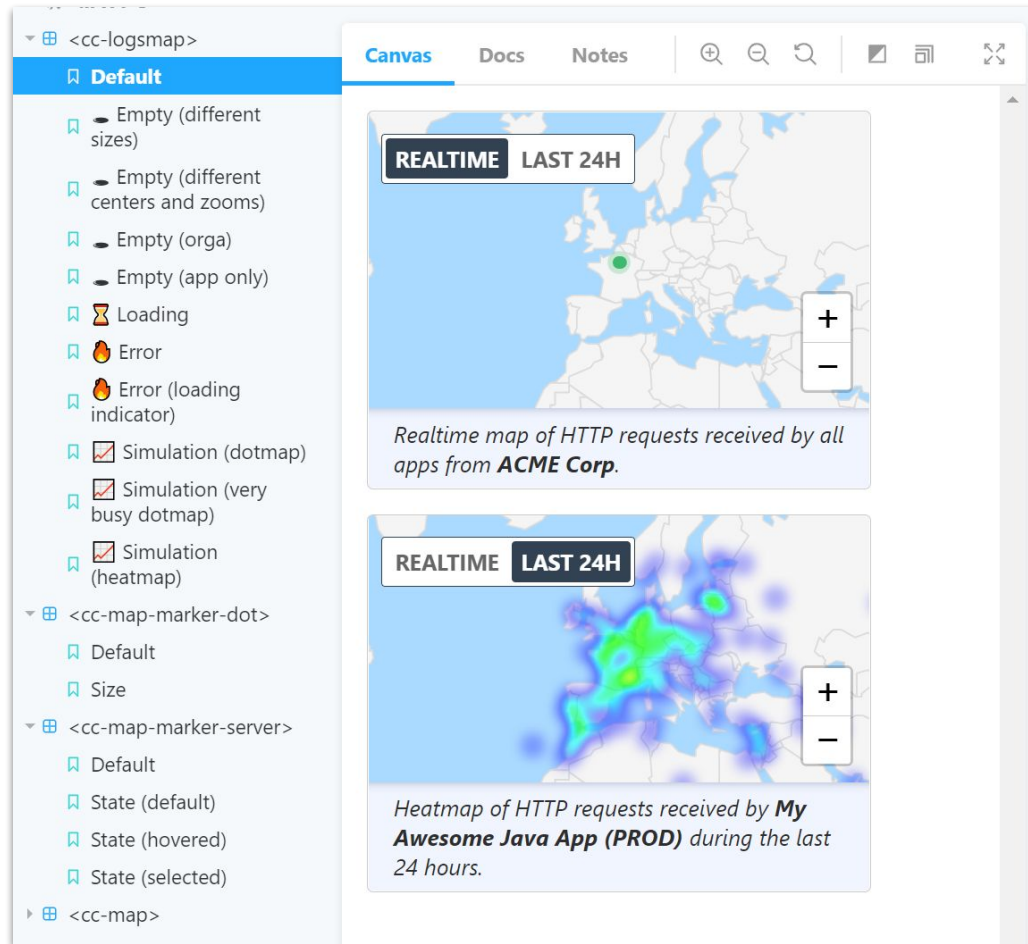
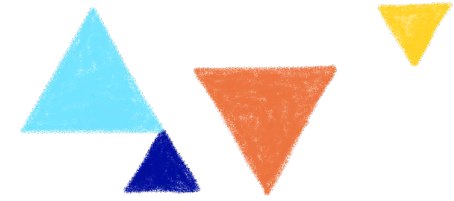
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque feugiat dui at leo porta dignissim. Etiam ut purus ultrices, pulvinar tellus quis, cursus massa. Mauris dignissim accumsan ex, at vestibulum lectus fermentum id. Quisque nec magna arcu. Quisque in metus sed erat sodales euismod eget id 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.



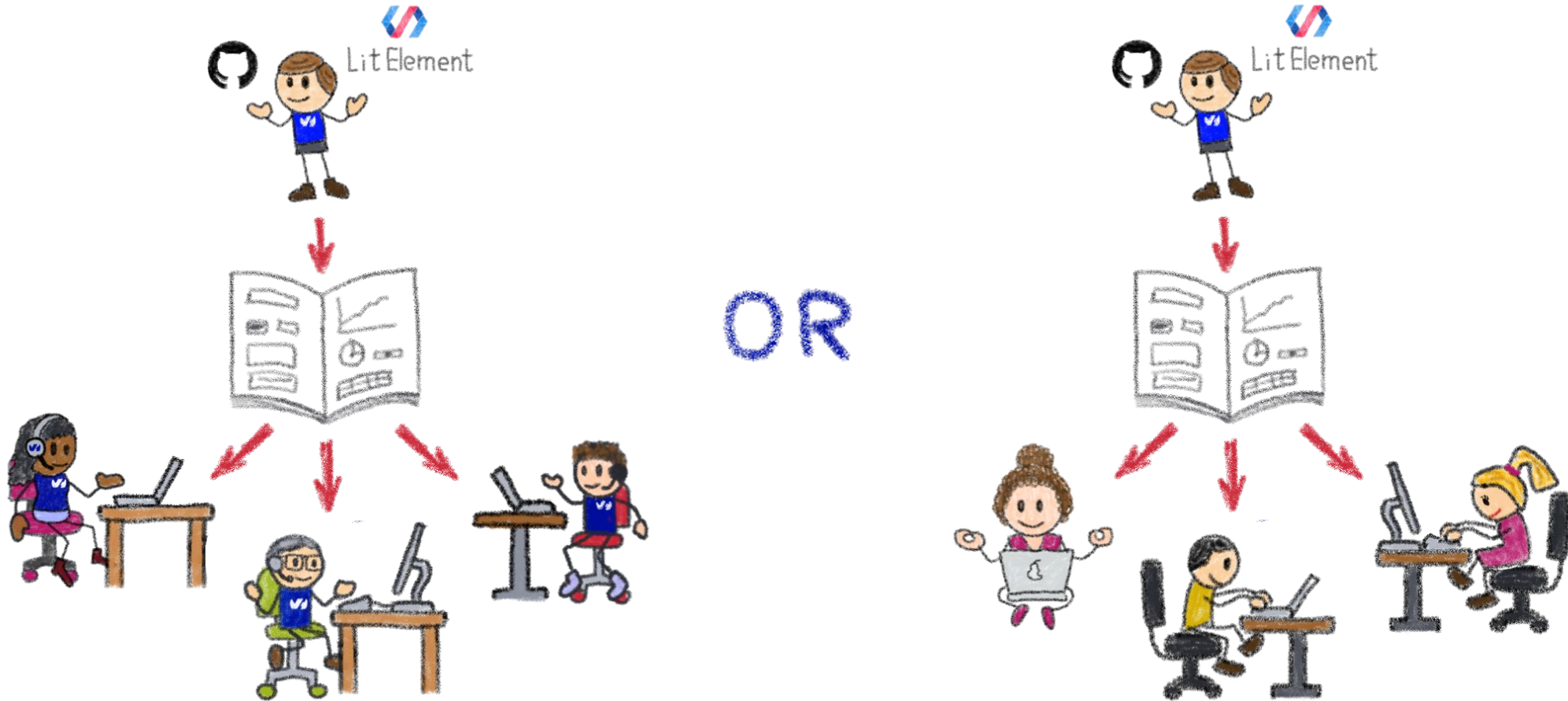
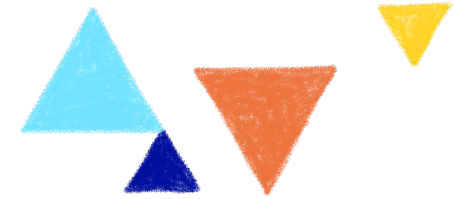
Capitalize on your atoms
Keep the flexibility and configurability

Big smart business components



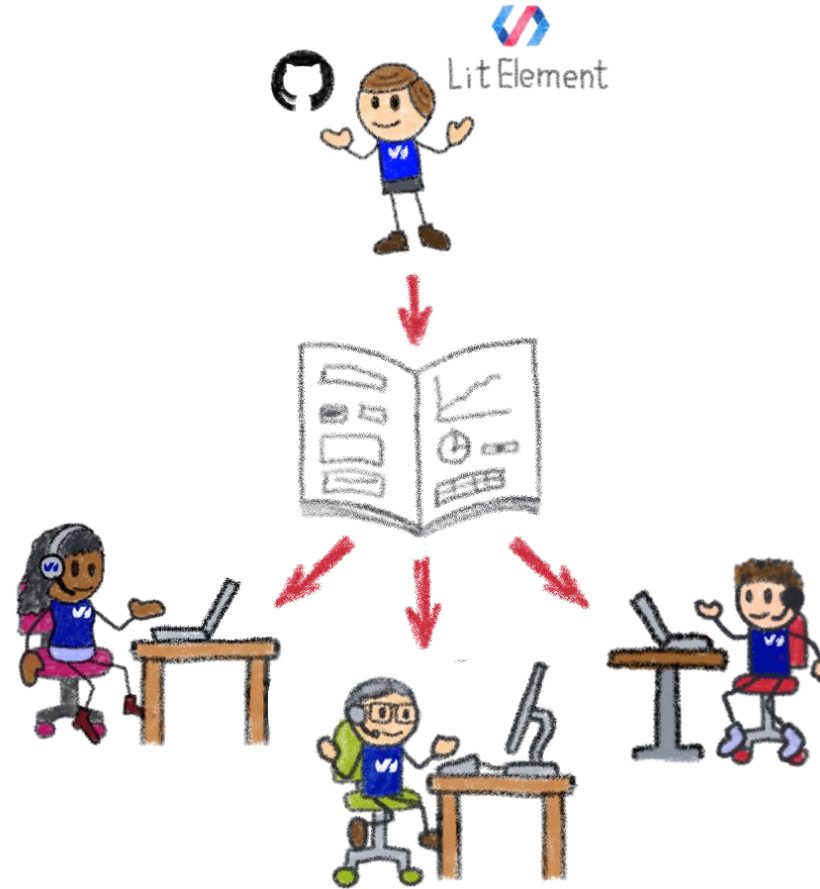
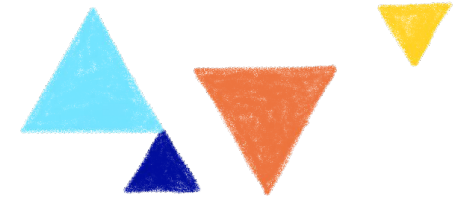
Encoding your business logic

Internal or external customers?



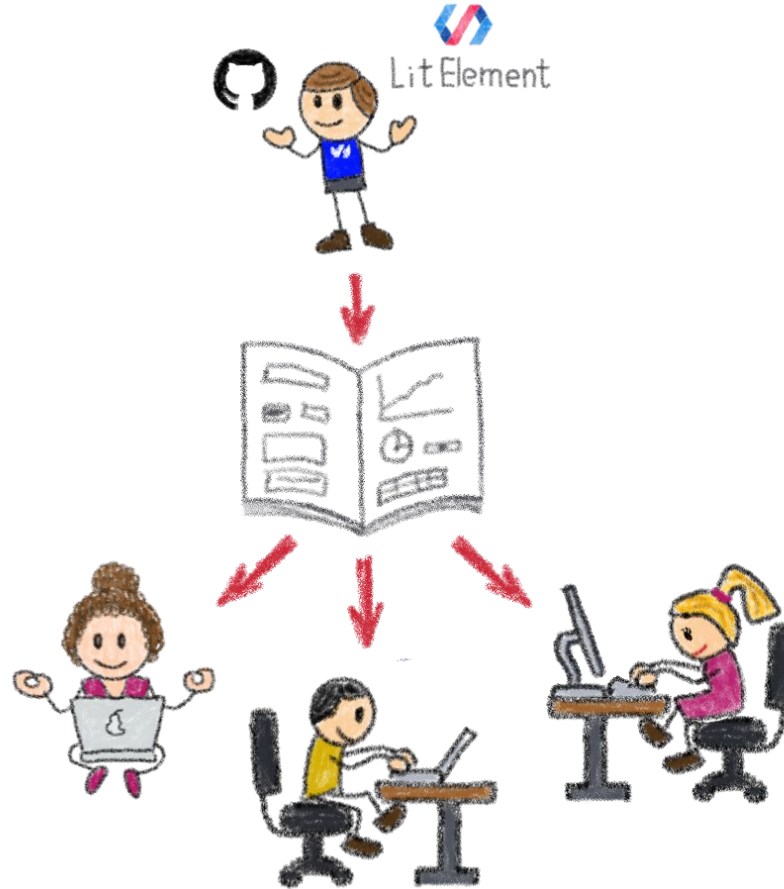
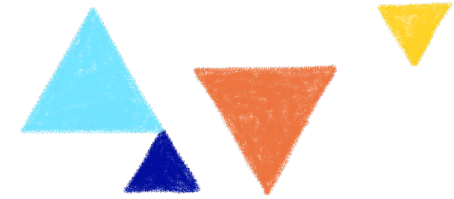
Who are your target users?

Internal customers need off-the-shelf components

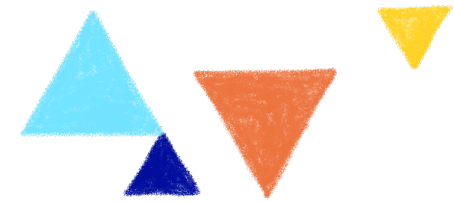


A well defined and coherent look-and-feel

External customers need to be able to tweak



Theming and customizing components

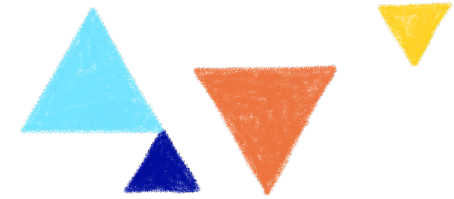


How to organize the catalog

Packages, imports and pragmatism



A single repository

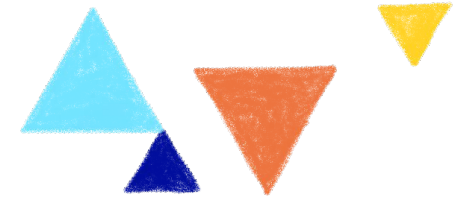


The screenshot shows the GitHub interface for the repository 'ing-bank / lion'. At the top, there's a search bar and navigation links for Pulls, Issues, Codespaces, Marketplace, and Explore. The repository name is 'ing-bank / lion' with 28 watches, 887 stars, and 144 forks. Below this, there are tabs for Code, Issues (27), Pull requests (16), Discussions, Actions, Projects, and Wiki. The main content area shows the file tree for the 'lion / packages /' directory on the 'master' branch. It lists several folders with their commit messages and dates:

Folder Name	Commit Message	Time Ago
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

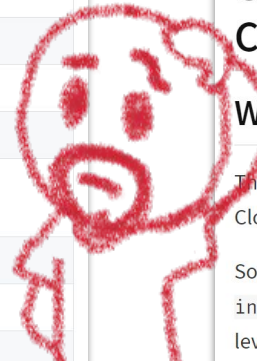
Two schools of thought



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 Music Products Pricing Documentation Community

npm Search packages Search Sign Up Sign In

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

@clevercloud/components
4.1.2 • Public • Published 2 months ago

Readme Explore BETA 10 Dependencies 0 Dependents 76 Versions

Collection of Web Components by Clever Cloud

Install

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

Weekly Downloads

232

Version	License
4.1.2	Apache-2.0

Unpacked Size	Total Files
1.31 MB	145

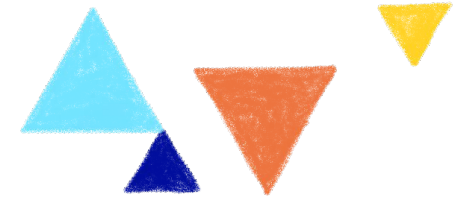
What is this?

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.

A packet per component or a global one

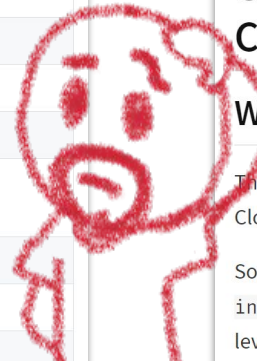
Two schools of thought



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 Music Products Pricing Documentation Community

npm Search packages Search Sign Up Sign In

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

@clevercloud/components
4.1.2 • Public • Published 2 months ago

Readme Explore BETA 10 Dependencies 0 Dependents 76 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

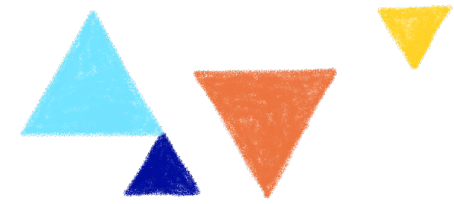
232

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



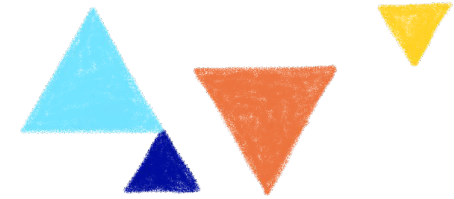
stencil



smart
HTML ELEMENTS

For different needs and sensibilities

Which ones to use?

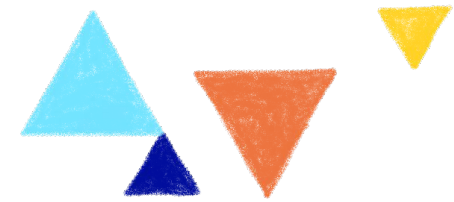


The screenshot shows the Clever Cloud website with a sidebar menu containing sections like HOME, README, CHANGELOG, CONTRIBUTING, and RELEASE. The main content area is titled "Collection of Web Components by Clever Cloud" and includes a "What is this?" section. Below this is a Storybook preview of the "Lion Web Components" project, which lists various components under "ATOMS" such as <cc-beta>, <cc-button>, <cc-dateline-relative>, <cc-expand>, <cc-flex-gap>, and <cc-img>. The Storybook page also features an introduction, a list of forms, and a "Demos" section.

The screenshot displays the "UI Components" page from the Ionic framework. It features a list of components on the left, including Action Sheet, Alert, Badge, Button, Card, Checklist, and Chip. The main content area is titled "UI Components" and explains that Ionic apps are made of high-level building blocks called Components. It lists several components including cards, lists, and tabs, and refers to the API Index for a complete list. Below the text are four cards, each representing a component: Action Sheet, Alert, Badge, and Card, with a brief description of each.

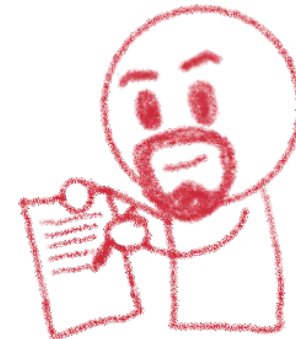


All are good, but these are popular favorites

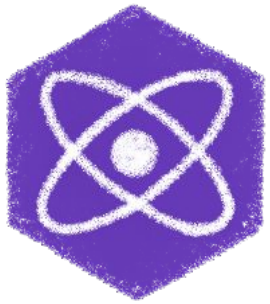
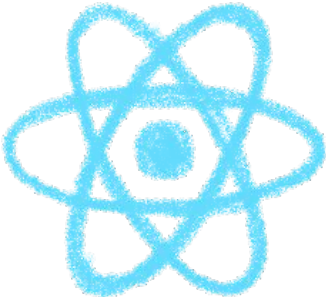
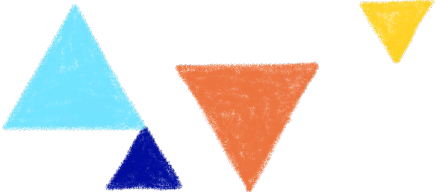


Driving-up adoption

Making devs use your components

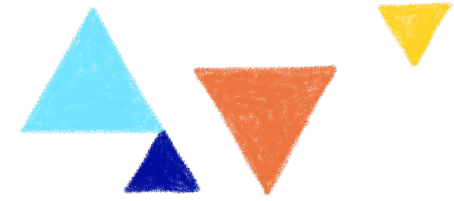


Think who are your target users



Users of any framework current or future...

They aren't used to your library



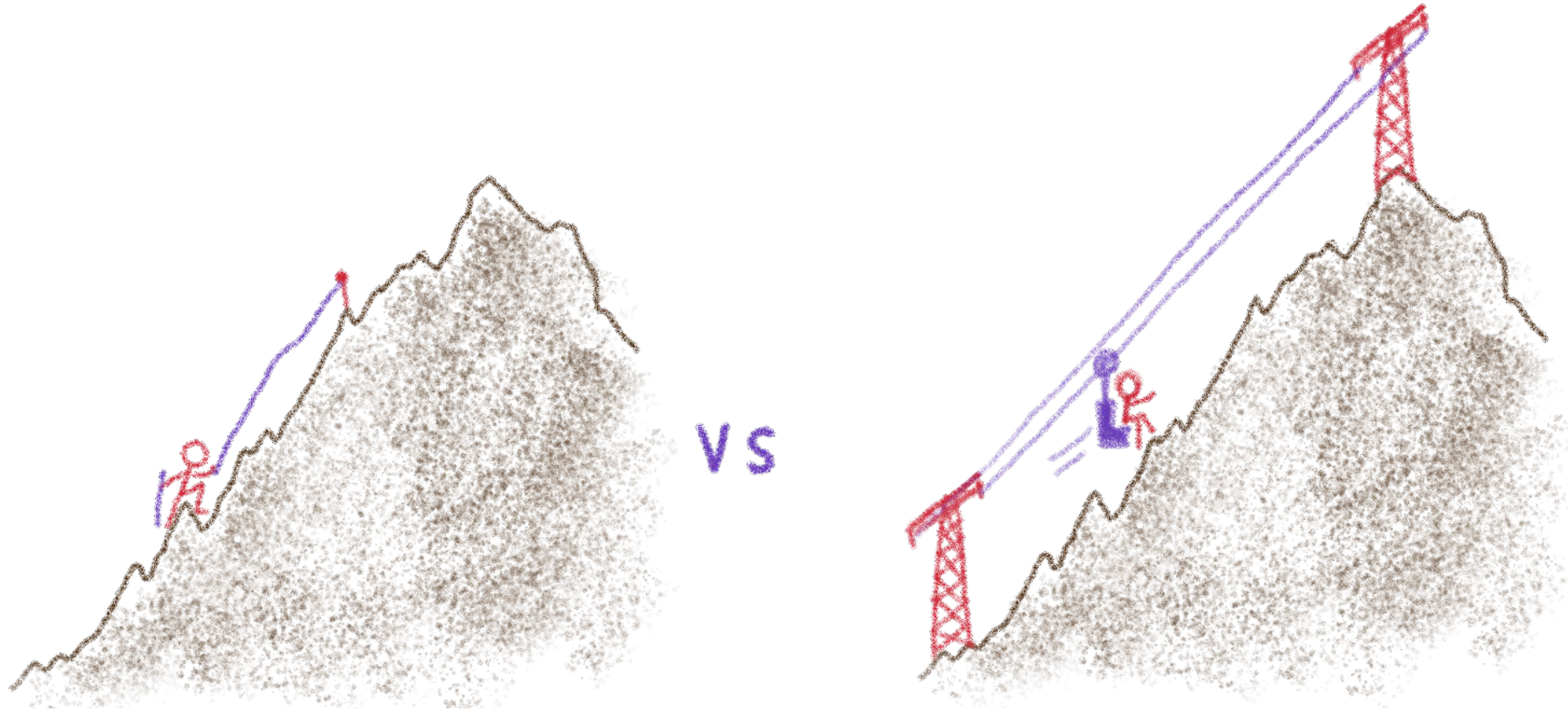
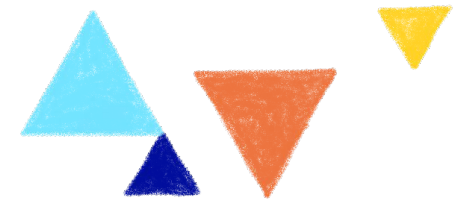
```
File Edit Selection View Go Debug Terminal Help my-component.tsx - Untitled (Workspace) - Visual Studio Code
EXPLORER
  my-component.tsx src/components/my-com...
  UNTITLED (WORKSPACE)
  sthlm.js
  .stencil
  dist
  node_modules
  src
    components / my-component
      my-component.css
      my-component.e2e.ts
      my-component.tsx
      readme.md
    utils
  components.d.ts
  index.html
  index.ts
  www
  .editorconfig
  .gitignore
  LICENSE
  package-lock.json
  package.json
  readme.md
  stencil.config.ts
  tsconfig.json
  OUTLINE
  NPM SCRIPTS
  WSL: Ubuntu 0 0 0 Ln 1, Col 1 Spaces: 2 UTF-8 LF TypeScript React 3.7.3

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
```



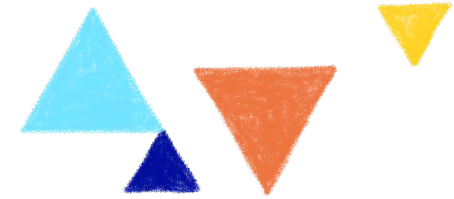
And they shouldn't need to be

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

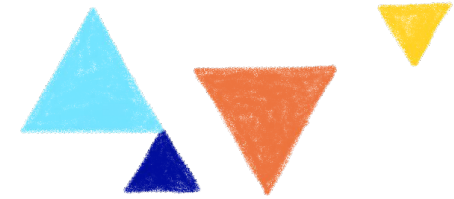
Use a Web Component

```
<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',
};

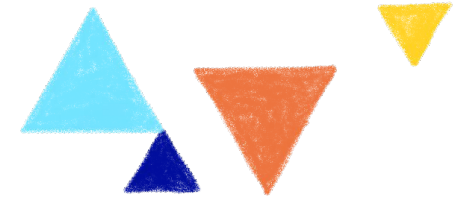
loadDefaultFeedbackMessages();

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 interface for 'Lion Web Components'. The left sidebar contains a navigation menu with sections 'INTRO' and 'FORMS'. Under 'INTRO', 'Lion Web Components' is selected, with sub-items 'Announcement' and 'Tabs Example'. Under 'FORMS', there are many sub-items 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 the following text:

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.

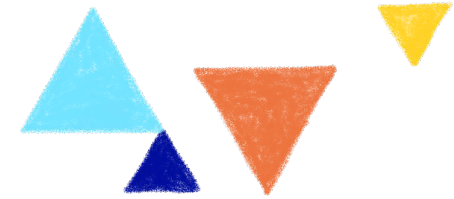
Please note: This project uses Yarn [Workspaces](#). If you want to run all demos locally you need to get [Yarn](#) and install all dependencies by executing `yarn install`.

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



The screenshot shows the Clever Cloud documentation interface. 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 (<cc-beta>, <cc-button>, <cc-datetime-relative>, <cc-expand>, <cc-flex-gap>, <cc-img>, <cc-input-text>). The main content area displays the title 'Web Components guidelines at Clever Cloud' and a sub-header 'General rules and reminders'. Below this is a list of guidelines.

Web Components guidelines at Clever Cloud

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

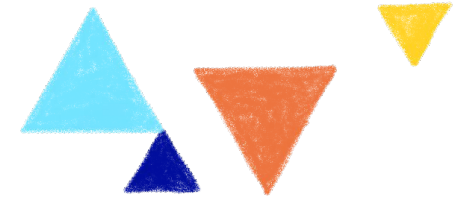
General rules and reminders

- 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



The screenshot shows the Clever Cloud documentation interface. 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 (<cc-beta>, Default, With tiles, With map, <cc-button>, <cc-datetime-relative>, <cc-expand>). The main content area is titled 'How to translate and localize?' and includes a welcome message, a section for 'OBSERVATIONS' with a list of points, and a section for 'As a user' with a list of steps.

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

