

Stencil

The Then, the Now and the Future



Manu Mtz.-Almeida

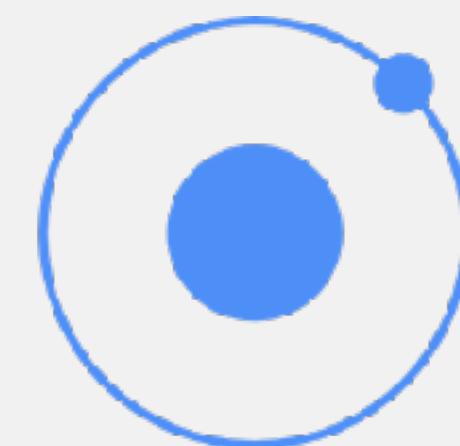
Software Engineer on the
Ionic Framework Team

 @manucorporat

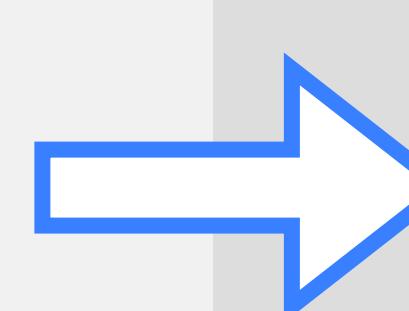
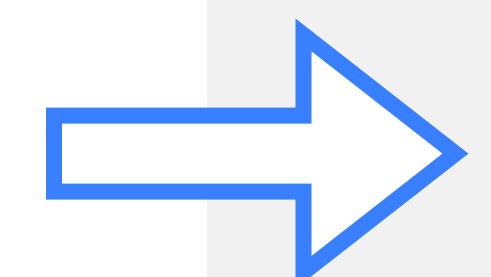
 @manucorporat



“ stencil is a toolchain for building reusable, scalable Design Systems, based in future-proof web standards. ”



ionic



Ember

Vue

React

Angular

Svelte





```
<script type="module" src="https://cdn.jsdelivr.net/npm/@ionic/core/dist/ionic/ionic.esm.js"></script>
<script nomodule src="https://cdn.jsdelivr.net/npm/@ionic/core/dist/ionic/ionic.js"></script>

<ion-button>Button</ion-button>
<ion-range min="0" max="10" value="2"></ion-range>
```



```
import { NgModule } from '@angular/core';
import { IonicModule } from '@ionic/angular';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';

@NgModule({
  declarations: [
    AppComponent,
  ],
  imports: [
    IonicModule.forRoot(),
    AppRoutingModule,
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```



```
import React from 'react';

import { IonButton, IonIcon, IonContent } from '@ionic/react';

export const Page = () => (
  <IonContent>

    <IonButton>
      <IonIcon slot="start" name="rocket" />
      Default
    </IonButton>

  </IonContent>
);

ReactDOM.render(<Page />, document.getElementById('app'));
```



```
import Vue from 'vue';
import Ionic from '@ionic/vue';
import App from './App.vue';

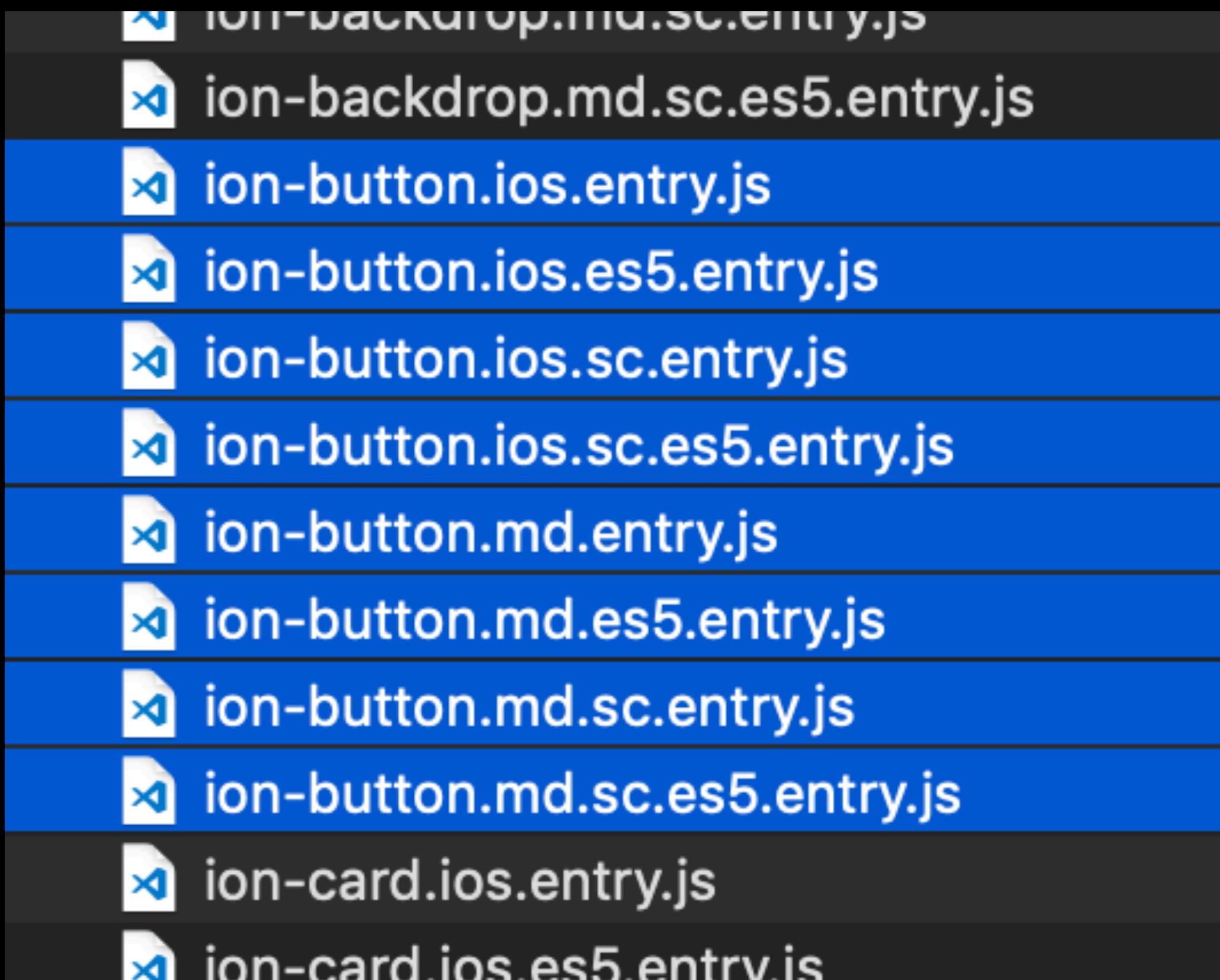
Vue.use(Ionic);

new Vue({
  render: h => h(App)
}).$mount('#app');
```

Output Targets

- Bindings
- SSR
- Docs
- Lazy loading
- Raw Web Components

Differential Builds



- Build with iOS and MD styles
- Build in ESM (modern)
- Build in ES5 (legacy, IE11)
- Build SC: browser that doesn't support native shadow-dom

Hot Module Replacement

Service Worker generation

Deep TS integration

Prerendering



Fast incremental builds

Familiar API

Docs generation

Visual-diff testing

E2E testing

Docs generation

The screenshot shows the Ionic Framework documentation for the 'ion-radio' component. At the top, there's a code editor window displaying a configuration file:

```
export const config = {
  outputTargets: [
    {
      type: 'docs',
    }
  ],
};
```

Below the code editor is the main content area for the 'ion-radio' component. It features a title 'ion-radio', a sidebar with 'CONTENTS' and links to 'Usage', 'Properties', 'Events', and 'CSS Custom Properties', and a main text section. The main text describes the usage of radios, mentioning they can be used alone or in groups, and how to check them programmatically.

On the right side of the main content area, there's a screenshot of a mobile device displaying the 'Radio' component. The device screen shows a list of options under three sections: 'Fruits (Group w/ values)', 'Extra Pizza Topping (Group w/ no values)', and 'Veggies (Group w/ allow empty)'. Under each section, there are radio buttons labeled with fruit names, toppings, and vegetable names respectively. Some radio buttons are checked, while others are not. At the bottom of the device screen, there are buttons for 'Checked', 'Disabled', 'Print', and 'Color'.

At the very bottom of the main content area, there's a 'Values:' section with the following code:

```
fruitRadio: grape
pizzaRadio: ion-rb-14
veggiesRadio: broccoli
```

The screenshot shows a 'readme.md' file for the 'ion-radio' component. The title is 'ion-radio'. The content starts with a general description of radios, stating they are generally used as a set of related options inside a group but can also be used alone. It mentions that pressing a radio will check it and that they can be checked programmatically by setting the 'checked' property.

It then describes how an 'ion-radio-group' can be used to group a set of radios. It states that when radios are inside a 'radio group', only one radio in the group will be checked at any time. Pressing a radio will check it and uncheck the previously selected radio, if there is one. If a radio is not in a group with another radio, then both radios will have the ability to be checked at the same time.

Properties

Property	Attribute	Description
checked	checked	If <code>true</code> , the radio is selected. Defaults to <code>false</code> .
color	color	The color to use from your application's color palette. Default options are: "primary", "secondary", "tertiary", "success", "warning", "danger", "light", "medium", and "dark". For more information on colors, see theming .
disabled	disabled	If <code>true</code> , the user cannot interact with the radio. Defaults to <code>false</code> .
mode	mode	The mode determines which platform styles to use. Possible values are: "ios" or "md".
name	name	The name of the control, which is submitted with the form data.
value	--	the value of the radio.

Events



Consolidation of the API

New compiler architecture

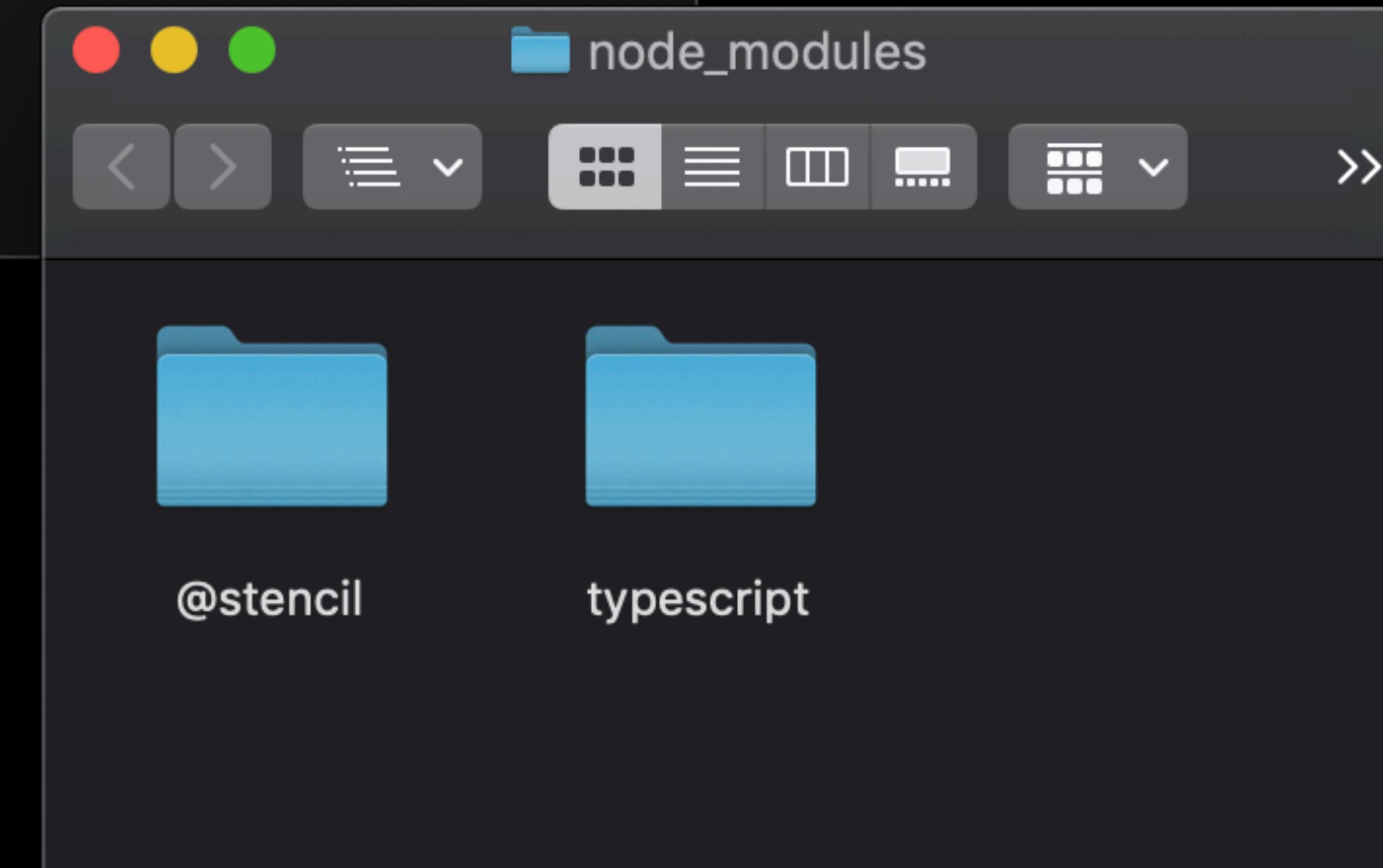
Zero dependencies

Faster/smaller runtime

Dependency free

```
[→ stencil-component-starter git:(master)] npm i
npm WARN stencil-starter-project-name@0.0.1 No repository field.

added 2 packages from 2 contributors and audited 2 packages in 1.072s
found 0 vulnerabilities
```



Components generated by
Stencil are dependency free

87 bytes
hello-world
(compressed)

Todos Stencil



What needs to be done?



Prepare talk for JSconf EU

Todos Stencil



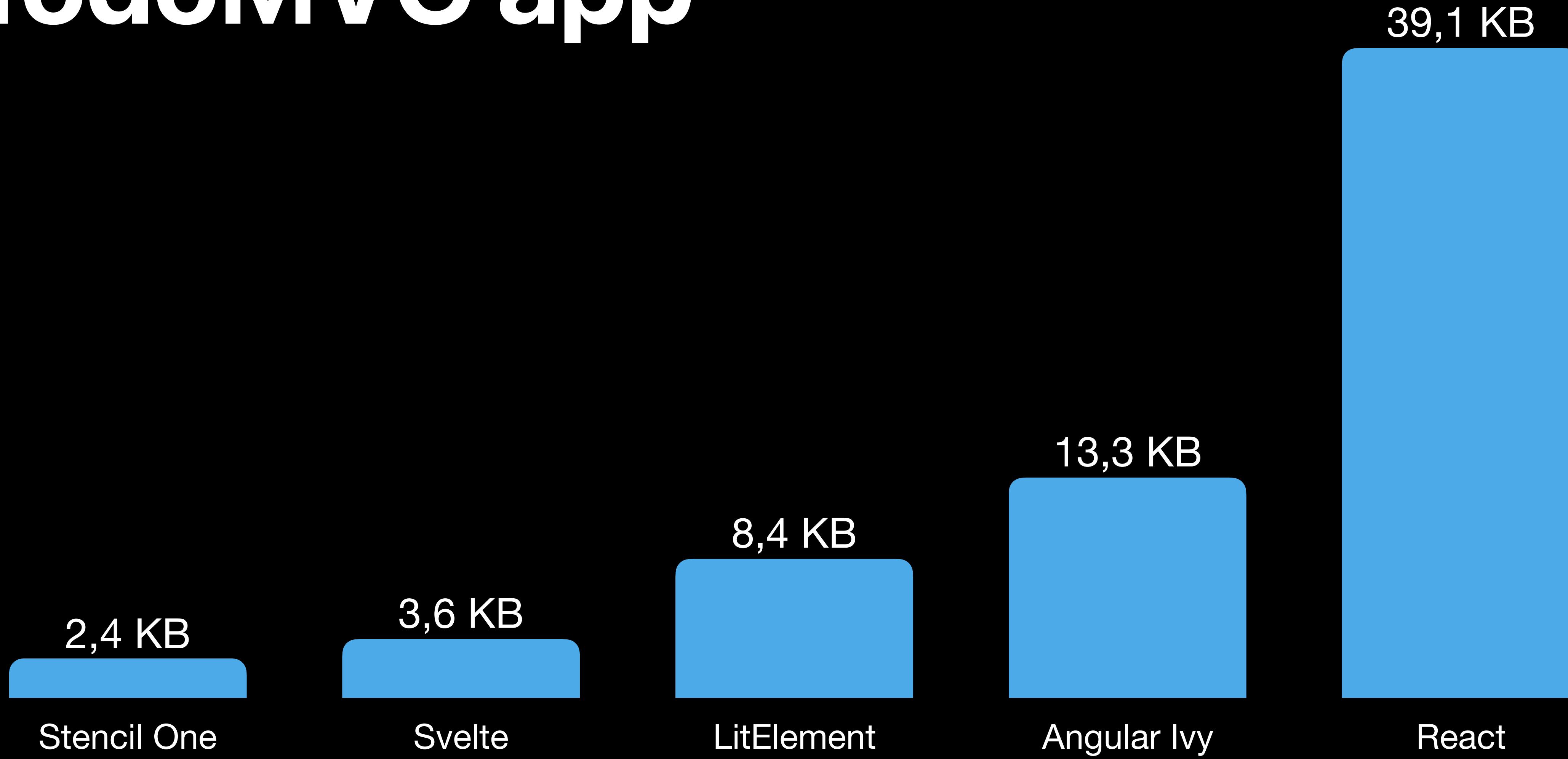
What needs to be done?

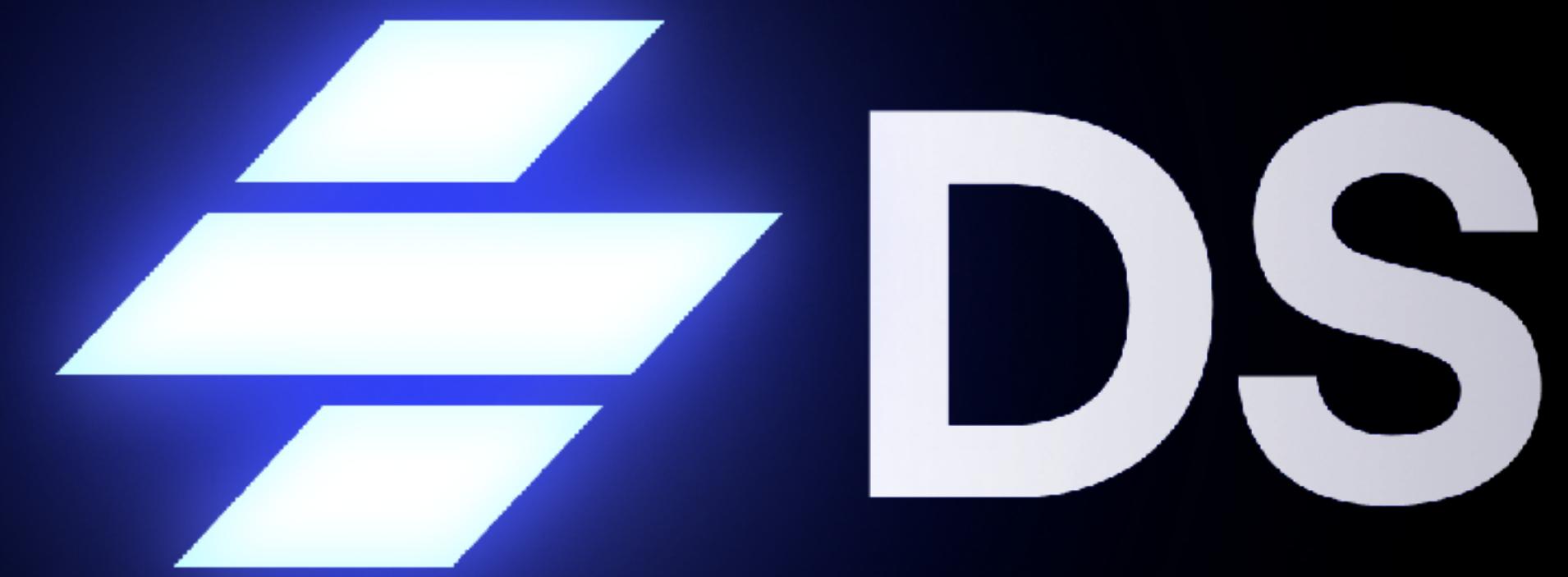
2.4 KB



Prepare talk for JSconf EU

TodoMVC app





- Support and Advisory
- Framework bindings
- Playground
- Automatic documentation site generation
- Usage tracking
- Visual Regression testing
- Per component versioning
- Training

Docs for your DS

System XYZ	0.0.1
OVERVIEW	
Getting Started	
Design Tokens	
Changelog	
COMPONENTS	
Badge	
Button	
Card	
Checkbox	
Code	
Container	
Dialog	
Grid	
Heading	
Input	

Getting Started

This is the description for the page, pulled from the data. It can be edited in the markdown files.

Installing lorem ipsum

System XYZ can be installed using the following command:

```
npm install @evilcorp/systemxyz
```

Installing dolor sit amet

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris:

```
npm install @evilcorp/component
```

Docs for your DS

System XYZ 0.7.8

OVERVIEW

- Getting Started
- Design Tokens
- Component Status

COMPONENTS

- Badge
- Button
- Card
- Checkbox
- Code
- Container
- Dialog
- Grid
- Heading
- Input

Card

Cards are a standard piece of UI that serves as an entry point to more detailed information. A card can be a single component, but is often made up of some header, title, subtitle, and content.

Here's a small text description for the card component. Nothing more, nothing less.

PROPS

- button
- disabled

color

href

type

- button
- reset
- submit

OVERVIEW

[Getting Started](#)[Design Tokens](#)[Component Status](#)

COMPONENTS

[Badge](#)[Button](#)[Card](#)[Checkbox](#)[Code](#)[Container](#)[Dialog](#)[Grid](#)[Heading](#)[Input](#)[Link](#)[Menu](#)[Pill](#)[Radio](#)[Select](#)

Card

Cards are a standard piece of UI that serves as an entry point to more detailed information. A card can be a single component, but is often made up of some header, title, subtitle, and content.

Here's a small text description for the card component. Nothing more, nothing less.

PROPS

button

disabled

color

undefined

href

undefined

type

button reset submit

Copy

```
<ds-card>
  <ds-card-content>
    <p>Here's a small text description for the card component. Nothing more, nothing less.</p>
  </ds-card-content>
</ds-card>
```

Badge

Button

Card

Checkbox

Code

Container

Dialog

Grid

Heading

Input

Link

Menu

Pill

Radio

Select

Slider

Switch

Text Area

Toggle

Tooltip

Here's a small text description for the card component. Nothing more, nothing less.

PROPS

button

disabled

color

undefined

href

undefined

type

button reset submit

 Copy

```
<ds-card>
  <ds-card-content>
    <p>Here's a small text description for the card component. Nothing more, nothing less.</p>
  </ds-card-content>
</ds-card>
```

Usage

Basic Example

Basic usage for the card component is quite simple. To use a card, follow this structure:

CONTENTS

[Usage](#)

Properties

CSS Custom Properties

Here's a small text description for the card component. Nothing more, nothing less.

Visual Diff testing

The screenshot shows a visual diff testing interface comparing two versions of an Ionic component, specifically the `Item Divider`.

Left Column: Shows various divider configurations:

- Divider by itself
- Divider in List
- Danger Divider in List** (highlighted in red)
- Divider in Item Group
- Secondary Divider in Item Group** (highlighted in cyan)
- Secondary header
Plain Ol' div with some text
- Item Divider Danger Span
- Multiline text that should wrap when it is too long to fit on one line in the item. Attribute on .item

Middle Column: Shows the same configurations as the left column, but with styling changes:

- DIVIDER BY ITSELF
- DIVIDER IN LIST
- DANGER DIVIDER IN LIST** (highlighted in red)
- DIVIDER IN ITEM GROUP
- SECONDARY DIVIDER IN ITEM GROUP** (highlighted in cyan)
- Secondary header
Plain Ol' div with some text
- ITEM DIVIDER DANGER SPAN
- Multiline text that should wrap when it is too long to fit on one line in the item. Attribute on .item

Right Column: Shows the differences between the two columns. The differences are highlighted in orange.

Diff	3974d7ff
Mismatched Pixels	248188
Mismatched Ratio	0.1939
Device	iPhone
Width	400
Height	800
Device Scale Factor	2
Left Preview	HTML
Right Preview	HTML
Description	item: dividers

Bottom Row: Shows the primary and secondary color palettes for both dark and light modes.

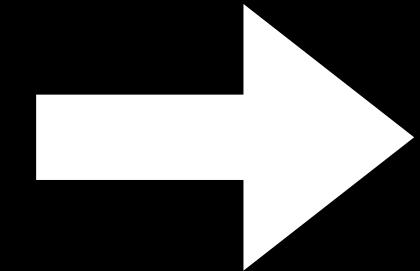
Color	Dark	Light
Primary	Primary h5	LIGHT
Secondary	DARK PRIMARY H5	PRIMARY

Roadmap

- CSS modules
- Web Workers
- Compiler API
- Faster Incremental Build
- New “Custom Elements” output
- Source maps
- Forms

CSS modules

```
● ● ●  
@Component({  
  tag: 'my-cmp',  
  stylesUrl: './my-component.css'  
)  
export MyComponent {}
```



```
● ● ●  
import styles from './my-component.css';  
  
@Component({  
  tag: 'my-cmp',  
  styles  
)  
export MyComponent {}
```

CSS modules

NO BREAKING CHANGES

```
...  
@Component({  
  tag: 'my-cmp',  
  stylesUrl: './my-component.css'  
})  
export MyComponent {}
```

```
import styles from './my-component.css';  
@Component({  
  tag: 'my-cmp',  
  styles  
})  
export MyComponent {}
```

CSS modules



```
import md from './my-component.md.css';
import ios from './my-component.ios.css';

@Component({
  tag: 'my-cmp',
  styles: [
    md,
    ios
  ]
})
export MyComponent {}
```

CSS modules



```
import globalStyles from '../global.css';
import styles from './my-component.css';

@Component({
  tag: 'my-cmp',
  styles: [
    globalStyles,
    styles
  ]
})
export MyComponent {}
```

CSS modules



```
import styles from './my-component.css';

// Dynamically modify styles
styles.insertRule('#blanc { color: white }', 0);

@Component({
  tag: 'my-cmp',
  styles
})
export MyComponent {}
```

CSS modules

- Single API for all CSS
- CSS unduplication out of the box
- Works better outside stencil
 - Build stencil outside stencil
- Extendable
- It will be eventually standarized
 - <https://github.com/w3c/webcomponents/issues/759>

Compiler API

Node CLI

Browser REPL

Custom

compiler.mjs

Same “script” works in Node and Browser

Compiler API

The screenshot shows a developer environment with two main panes. The top pane is a web browser window titled "Stencil REPL" displaying the "Hello Stencil" application. It has tabs for "/hello-world.tsx" (selected) and "/util.ts". On the right, there's a code editor with a dropdown menu set to "custom-element-next" showing the generated JavaScript code for the "HelloWorld" component. The bottom pane is a terminal window showing the build process for "core — npm start". The output includes logs from "stencil/core v0.0.0-stencil-dev", build steps like transpile, type checking, and bundling, and a message about it being a prerelease build. A yellow banner at the bottom right of the terminal says "UNDER CONSTRUCTION".

```
core — npm start — npm — node -v npm TMPDIR=/var/folders/vb/4xds70ts5js1l1z954k2hnr0...  
[24:33.3] @stencil/core v0.0.0-stencil-dev ⚡  
[ WARN ] This is a prerelease build, undocumented changes might happen  
at any time. Technical support is not available for prereleases, but  
any assistance testing is appreciated.  
[24:33.8] build, ionic, dev mode, started ...  
[24:34.1] transpile started ...  
[24:35.2] transpile finished in 1.14 s  
[24:35.3] type checking started ...  
[24:35.3] copy started ...  
[24:35.3] generate styles started ...  
[24:35.3] bundling components started ...  
[24:42.1] type checking finished in 6.85 s  
[24:42.1] copy finished (1060 files) in 6.87 s  
[24:42.6] generate styles finished in 7.30 s  
[24:43.0] bundling components finished in 7.77 s  
[24:43.5] dev server: http://localhost:3333/  
[24:43.5] build finished, watching for changes... in 9.65 s
```

```
custom-element-next  
/hello-world.js  
  
const getText = () => {  
  const text = 'Hello World';  
  return text;  
};  
  
const HelloWorld = class extends HTMLElement {  
  render() {  
    const val: string = getText();  
    return val;  
  }  
  connectedCallback() {  
    this.textContent = this.render();  
  }  
};  
export { HelloWorld };
```

UNDER CONSTRUCTION

Compiler API

The screenshot shows a developer environment with two main panes. The top pane is a web browser window titled "Stencil REPL" displaying the "Hello Stencil" application. It has tabs for "/hello-world.tsx" (selected) and "/util.ts". On the right, there's a code editor with a dropdown menu set to "custom-element-next" showing the generated JavaScript code for the "HelloWorld" component. The bottom pane is a terminal window showing the build process for "core — npm start". The output includes logs from "stencil/core v0.0.0-stencil-dev", build steps like transpile, type checking, and bundling, and a message about it being a prerelease build. A yellow banner at the bottom right of the terminal says "UNDER CONSTRUCTION".

```
core — npm start — npm — node -v npm TMPDIR=/var/folders/vb/4xds70ts5js1l1z954k2hnr0...  
[24:33.3] @stencil/core v0.0.0-stencil-dev ⚡  
[ WARN ] This is a prerelease build, undocumented changes might happen  
at any time. Technical support is not available for prereleases, but  
any assistance testing is appreciated.  
[24:33.8] build, ionic, dev mode, started ...  
[24:34.1] transpile started ...  
[24:35.2] transpile finished in 1.14 s  
[24:35.3] type checking started ...  
[24:35.3] copy started ...  
[24:35.3] generate styles started ...  
[24:35.3] bundling components started ...  
[24:42.1] type checking finished in 6.85 s  
[24:42.1] copy finished (1060 files) in 6.87 s  
[24:42.6] generate styles finished in 7.30 s  
[24:43.0] bundling components finished in 7.77 s  
[24:43.5] dev server: http://localhost:3333/  
[24:43.5] build finished, watching for changes... in 9.65 s
```

```
custom-element-next  
/hello-world.js  
  
const getText = () => {  
  const text = 'Hello World';  
  return text;  
};  
  
const HelloWorld = class extends HTMLElement {  
  render() {  
    const val: string = getText();  
    return val;  
  }  
  connectedCallback() {  
    this.textContent = this.render();  
  }  
};  
export { HelloWorld };
```

UNDER CONSTRUCTION

Compiler API



```
import { createCompiler } from './compiler.mjs';

// Create a compiler with some user config.
const compiler = await createCompiler(CONFIG);

// Single build
const buildResults = await compiler.build();

// Watch build
const watcher = await compiler.createWatcher();
watcher.on('buildFinish', results => {
  console.log(results);
});
watcher.start();
```

Fast Incremental Builds

- Typescript 3.6 incremental build
- All builds are incremental (even the first one)
- Typechecking is incremental
- Transpilation + type checking
- From 10seconds to ~100ms (in ionic/core)

New “Custom Elements”

- Web Components extending from
HTMLElement
- No lazy loading
- No self-register
- Works out of the box with WebPack / Rollup

New “Custom Elements”

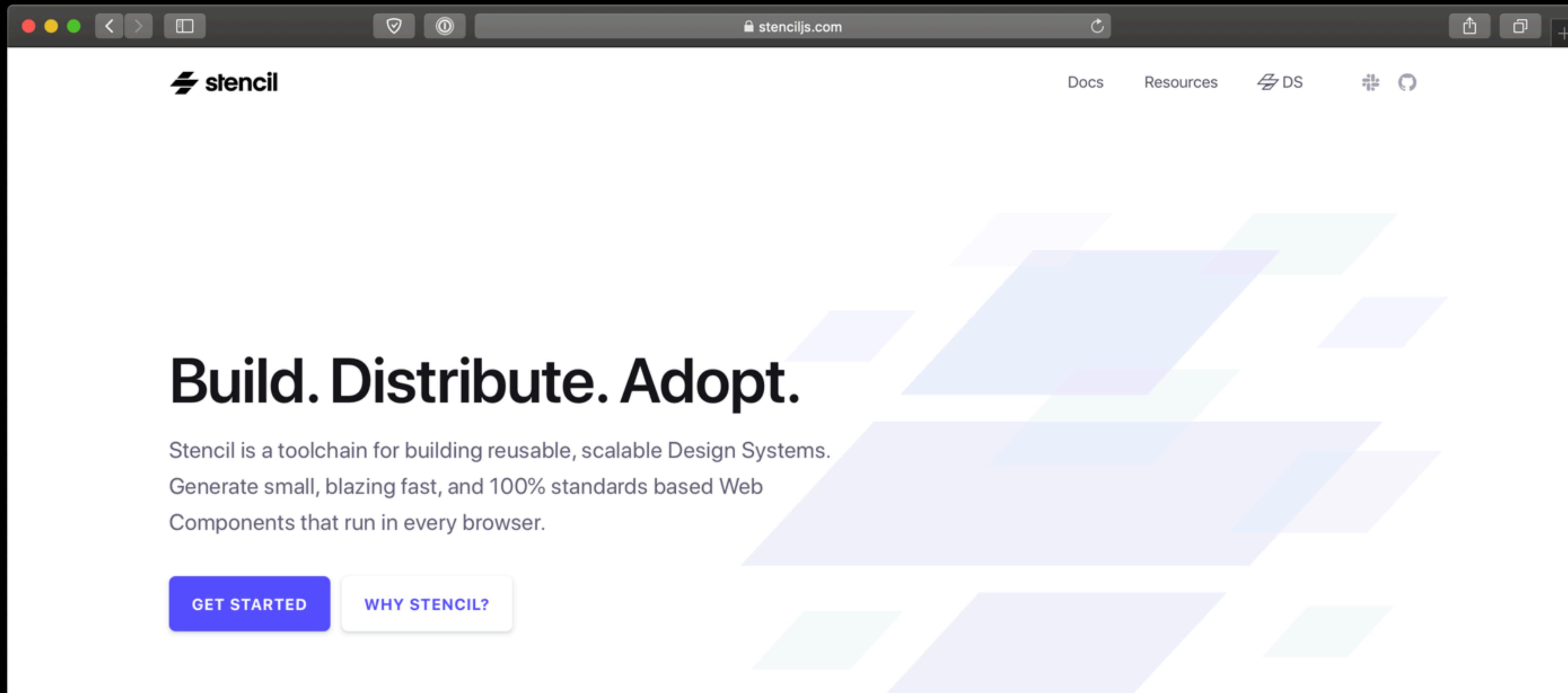


```
import { IonToggle } from './ionic';

customElements.define('my-toggle', IonToggle);
```

stenciljs.com

\$ npm init stencil



Thank You

@manucorporat

