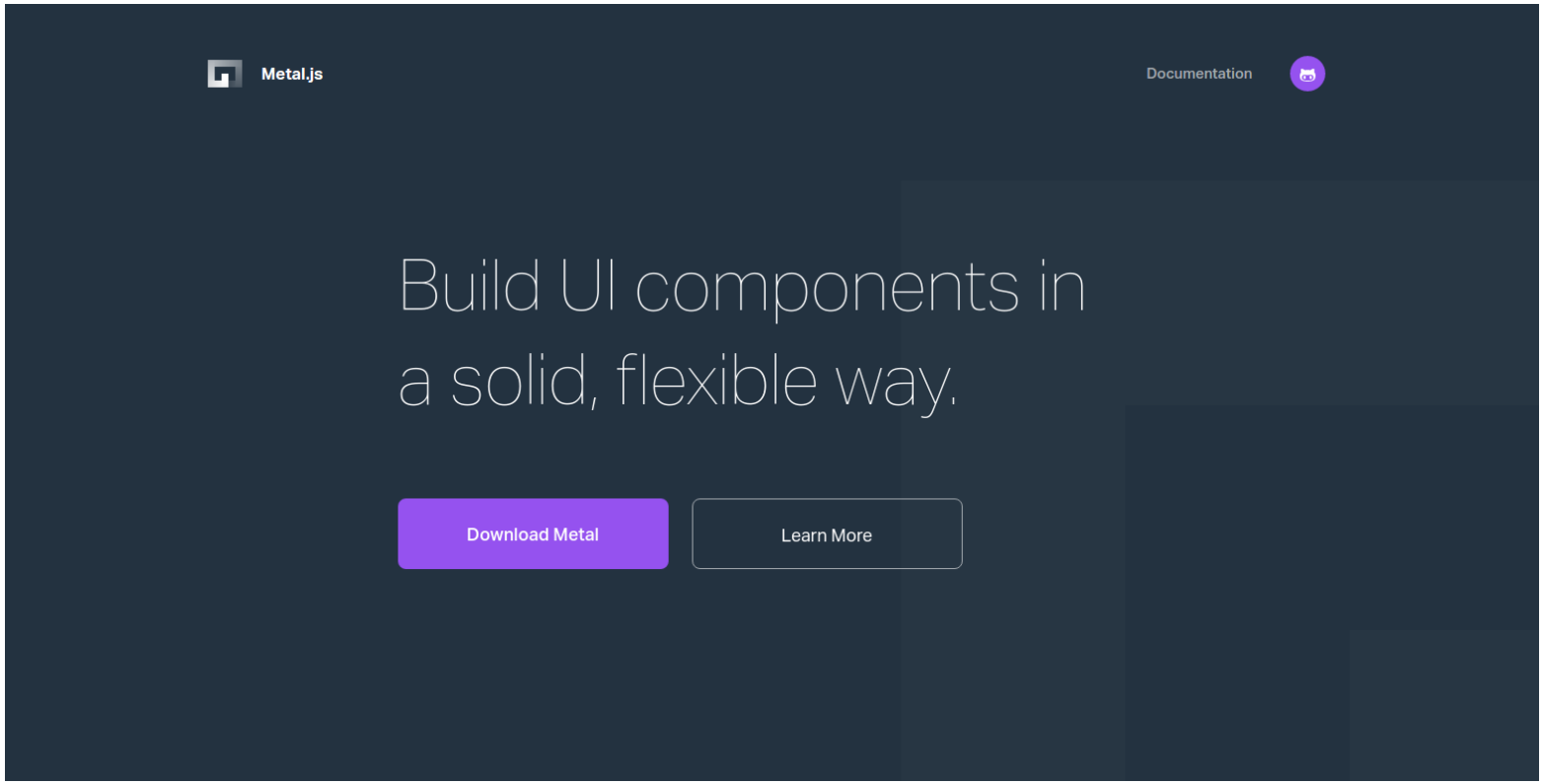
BUILDING COMPONENTS: METAL.JS

BUILD UI COMPONENTS IN A SOLID, FLEXIBLE WAY

- Metal.js is a lightweight, easy-to-use JavaScript framework that integrates with templating languages to help you create UI Components.

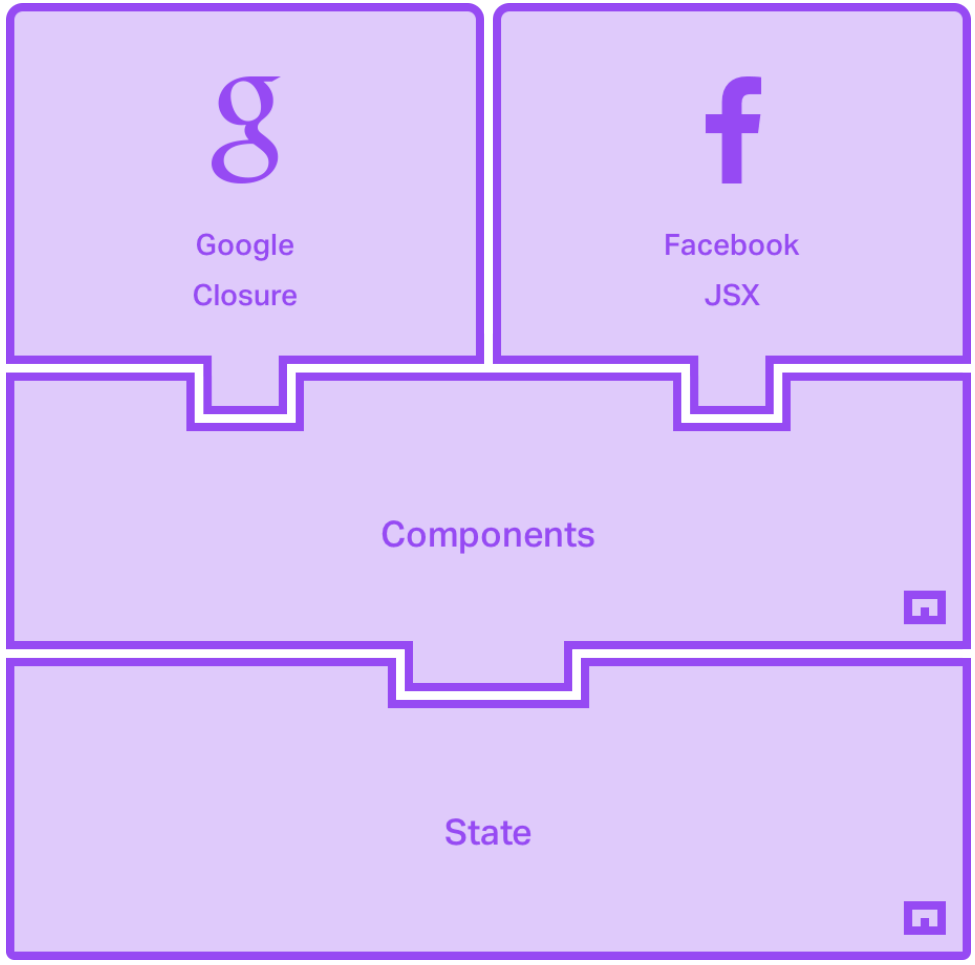


ARCHITECTURE

- Metal.js's main classes are *State* and *Component*.
- *Component* actually extends from *State*, so it contains all its features.
- The main difference between the two is that *Component's* extra features are related to rendering.
- If your module doesn't do any rendering, you could just use *State* directly.
- *Component* will work better for you if your module needs rendering logic.
- Many people have their favorite way of dealing with rendering logic.
- Some prefer to use template languages that completely separate the rendering logic from the business logic, while others like to keep both close together in the same file.
- Metal.js doesn't force developers to go with only one option.

STRUCTURE VISUALIZED

- Here's a visualization of the structure:



TEMPLATES

- Metal.js offers integration points with both Closure Templates from Google (Soy Templates) and JSX from Facebook.
- It's possible to add more options, as the Rendering Layer is customizable.
- A Soy Template in Metal.js may look like this:

```
{template .render}
  // ...
    <button onClick="{$close}" type="button" class="close">
  // ...
{/template}
```

AN ECMASCRIPT 2015 COMPONENT IN METAL.JS

- A Metal.js component written in *ECMAScript 2015* may look like this:

```
class MyComponent extends Component {
    created() {
        //do some things
    }

    disposed() {
        //do some other things
    }
}
```

- These just scratch the surface of the new features in ECMAScript 2015 that can be leveraged in Metal.js.

Notes: