Building Libraries with Vue CLI

Best practices and recommendations

@linus_borg
Vuejs Amsterdam
14.02. - 15.02. 2019

Building Libraries with Vue CLI

Best practices and recommendations

@linus_borg
Vuejs Amsterdam
14.02. - 15.02. 2019

whoami?

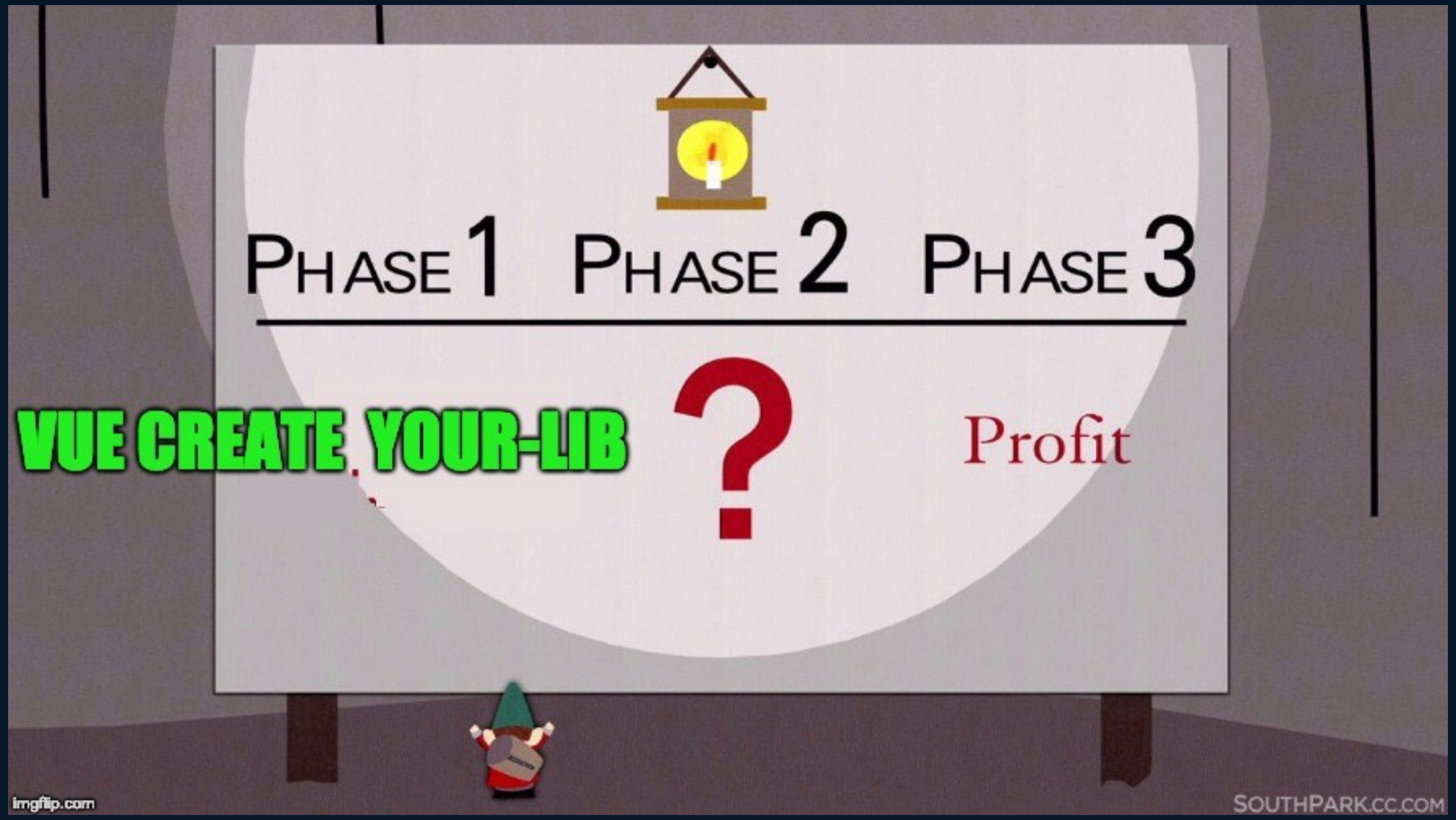
Thorsten Lünborg
(Linusborg)

Vue.js core team member

Forum-Question-Answerer

https://forum.vuejs.org

Github: linusborg
Twitter: @linus_borg


PHASE 1
PHASE 2
PHASE 3
VUE CREATE YOUR-LIB
?
Profit
imgflip.com
SOUTHPARK.CC.COM

- Vue CLI Project setup
- Enter: The entry file
- Weight Watchers

What is Vue CLI?

if you don’t know, I know you skipped some talks …

Project setup

with Vue CLI

Default setup is for apps

```
yarn build
```

```
yarn run v1.12.3
$ vue-cli-service build

⠸  Building for production...

 DONE  Compiled successfully in 7140ms                                    18:21:32

  File                                  Size               Gzipped

  dist/js/chunk-vendors.06c676b8.js     82.11 KiB          29.74 KiB
  dist/js/app.d3d5fb95.js               4.60 KiB           1.65 KiB
  dist/css/app.e2713bb0.css             0.33 KiB           0.23 KiB

  Images and other types of assets omitted.

 DONE  Build complete. The dist directory is ready to be deployed.
 INFO  Check out deployment instructions at https://cli.vuejs.org/guide/deployment.html

✨  Done in 10.44s.
```

But Vue CLI can build libraries as well!

```
yarn build --target lib –name YourLib src/index.js
```

```
yarn run v1.12.3
$ vue-cli-service build --target lib --name YourLib src/index.js

ℹ  Building for production as library (commonjs,umd,umd-min)...


 DONE  Compiled successfully in 4433ms                                    18:28:08

  File                      Size                    Gzipped

  dist/YourLib.umd.min.js   22.25 KiB               7.51 KiB
  dist/YourLib.umd.js       59.47 KiB               14.07 KiB
  dist/YourLib.common.js    59.00 KiB               13.89 KiB
  dist/YourLib.css          0.17 KiB                0.13 KiB

  Images and other types of assets omitted.

✨  Done in 6.04s.
```

1. **Adjust files & folders**
2. **Customise build scripts in** `package.json`
3. **Add library-specific fields to** `package.json`

Adjusting build scripts

`/src` contains boilerplate for an app - what to do with it?

```
mv ./src ./demo
```

recycle it as a demo app

```
touch ./src/index.js
```

our lib’s entry file

./demo/main.js

```
import Vue from 'vue'
import App from './App.vue'

import YourLib from '../src'
Vue.use(YourLib)

Vue.config.productionTip = false

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

„Whats the perfect folder structure?“

https://react-file-structure.surge.sh/

Chrome Datei Bearbeiten Anzeigen Verlauf Lesezeichen Personen Fenster Hilfe 100 % So. 18:11

Scalable file structure for Reac

https://react-file-structure.surge.sh

move files around until it feels right

Customise build scripts

./package.json

```
"scripts": {
  "serve": "vue-cli-service serve",
  "build": "vue-cli-service build",
  "lint": "vue-cli-service lint",
  "test:e2e": "vue-cli-service test:e2e",
  "test:unit": "vue-cli-service test:unit"
},
```

./package.json

```
"scripts": {
  "serve": "vue-cli-service serve ./demo/main",
  "build:demo": "vue-cli-service build ./demo/main“,
  "build": "vue-cli-service build --target lib --name YourLib src/index.js",
  "lint": "vue-cli-service lint",
  "test:e2e": "vue-cli-service test:e2e",
  "test:unit": "vue-cli-service test:unit"
},
```

Add library-specific fields

./package.json

```
{
  "main": „dist/YourLib.umd.js“,

  "browser": „dist/YourLib.common.js",

  "unpkg": "dist/YourLib.umd.min.js",
  "jsDelivr": "dist/YourLib.umd.min.js",

  "files": ["dist", "src"],

  "peerDependencies": {
    "vue": "^2.5.22"
  },
}
```

export the UMD file as main, as it works everywhere

The „browser“ field overwrites „main“ in bundlers like webpack

Fields for popular CDN providers, so they serve the minified file by default

Add /src to allow people to include it directly *

Vue should be a peer dependency, not a direct dependency

Enter: the Entry File

Please come in!

Starting simple

/src/index.js

```
import HelloWorld from './components/HelloWorld.vue'

export default HelloWorld
```

- Import the component
- Export it

There’s more to take care of!

- **Deal with multiple components**
- **Turn it into a Vue Plugin (Vue.use)**
- **Make it customisable**
- **Auto-Install it when included directly in a Browser**

Dealing with multiple components

/src/index.js

```
import HelloWorld from './components/HelloWorld.vue'

export default HelloWorld
```

/src/index.js

```
import HelloWorld from './components/HelloWorld.vue'
import GoodbyeReality from './components/GoodbyeReality.vue'

export {
  HelloWorld,
  GoodbyeReality
}

// Optional default export
export default HelloWorld
```

some-app/src/main.js

```
import Vue from 'vue'
import  {
  HelloWorld,
  GoodbyeReality
} from 'your-lib'

Vue.component('HelloWorld', HelloWorld)
Vue.component('GoodbyeReality', GoodbyeReality)
```

```
import Vue from 'vue'
import  {
  HelloWorld,
  GoodbyeReality
} from 'your-lib'

Vue.component('HelloWorld', HelloWorld)
Vue.component('GoodbyeReality', GoodbyeReality)
```

Only import what you need

Name components how you like

Tedious with a lot of components

No defaults

No additional config possible

Turning it into a Vue Plugin

What is a Vue Plugin ?

> *A Vue.js plugin should expose an install method.*
>
> *The method will be called with the Vue constructor as the first argument, along with possible options:*

https://vuejs.org/v2/guide/plugins.html#Writing-a-Plugin

/src/index.js

```
import HelloWorld from './components/HelloWorld.vue'
import GoodbyeReality from './components/GoodbyeReality.vue'

function install(Vue, options = {}) {

  Vue.component('HelloWorld', HelloWorld)
  Vue.component('GoodbyReality', GoodbyReality)
}

// Export the plugin function
export default install
```

some-app/src/main.js

```
import Vue from 'vue'
import  YourLib from 'your-lib'

Vue.use(YourLib)
```

```
import Vue from 'vue'
import  YourLib from 'your-lib'

Vue.use(YourLib)
```

Easy Installation with 2 lines of code

No matter how many components

Component Names are hardcoded

Risk: naming conflicts

/src/index.js

```
function install(Vue, options = {}) {

  Vue.component('HelloWorld', HelloWorld)
  Vue.component('GoodbyReality', GoodbyReality)
}
```

hardcoded component name

⛔ **Anti-Pattern**

Making it customisable

/src/index.js

```
import HelloWorld from './components/HelloWorld.vue'
import GoodbyeReality from './components/GoodbyeReality.vue'

function install(Vue, options = {}) {

  Vue.component(options.HelloWorldName || 'HelloWorld', HelloWorld)
  Vue.component(options.GoodbyeRealityName || 'GoodbyReality', GoodbyReality)
}

// Export the plugin function
export default install
```

some-app/src/main.js

```
import Vue from 'vue'
import  YourLib from 'your-lib'

Vue.use(YourLib, {
  HelloWorldName: 'BetterWorld'
})
```

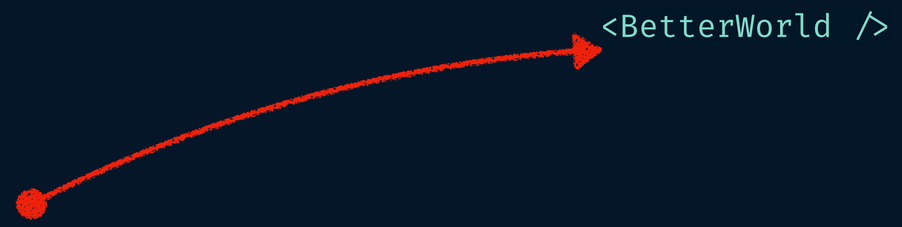
<BetterWorld />

/src/index.js

```
import HelloWorld from './components/HelloWorld.vue'
import GoodbyeReality from './components/GoodbyeReality.vue'

function install(Vue, options = {}) {

  Vue.component(options.HelloWorldName || 'HelloWorld', HelloWorld)
  Vue.component(options.GoodbyeRealityName || 'GoodbyReality', GoodbyReality)
}

// Export the plugin function
export default install

export {
  HelloWorld,
  GoodbyeReality
}
```

some-app/src/main.js

```
import Vue from 'vue'
import  {
  HelloWorld
} from 'your-lib'

Vue.component('BetterWorld', HelloWorld)
```



Making it even more customisable

/src/index.js

```
import StoreModule from './module'
import YourMixin from './module'

function install(Vue, {router, store, mixin } = {}) {

// Register components as before …
 router.beforeEach((to, from, next) => {
    next()
  })

  store.registerModule(moduleName, StoreModule)

  if (mixin) {
    Vue.mixin(YourMixin)
  }

}

// Export the plugin function
export default install

export {
  HelloWorld,
  GoodbyeReality,
  StoreModule,
  YourMixin,
}
```

Auto-Install when included directly in the browser

/src/index.js

```
import Vue from 'vue'
import HelloWorld from './components/HelloWorld.vue'
import GoodbyeReality from './components/GoodbyeReality.vue'

function install(Vue, options = {}) {

  Vue.component(options.HelloWorldName || 'HelloWorld', HelloWorld)
  Vue.component(options.GoodbyeRealityName || 'GoodbyReality', GoodbyReality)
}

// Export the plugin function
export default install

if (typeof window !== undefined
    && window.Vue
    && window.Vue === Vue) {
  install(window.Vue)
}

export {
  HelloWorld,
  GoodbyeReality
}
```

- Always export **all** components* individually
- Offer a way to install as a **plugin**
- Keep the plugin **customisable**
- **Auto-Install** in non-module environments

* components, directives, mixins, everthing

Weight Watchers

Babel Config

/babel.config.js

```
module.exports = {
  presets: ['@vue/app'],
}
```

- Based on @babel/preset-env
- preconfigured to inject polyfills and helpers
- Default settings best suited for an app, not a library

https://github.com/vuejs/vue-cli/tree/dev/packages/@vue/babel-preset-app

/babel.config.js

```
module.exports = {
  presets: [
    [
      '@vue/app',
      {
        useBuiltIns: false,
        polyfills: false,
      },
    ],
  ],
}
```

- Don’t bundle polyfills with your components
- Document required poly fills in your library’s docs
- The end user has to add these polyfills in their app

Externalise Dependencies

- Don’t bundle runtime dependencies (i.e. `lodash`)
- Why? Bundled dependencies can result in duplicated code
- Instead: tell webpack where to load them from when used

„externals“

💡

Vue CLI does this for the 'vue' package automatically

https://webpack.js.org/configuration/externals/

/vue.config.json

```
import _ from 'lodash'
```

```
module.exports = {
  lintOnSave: false,
  configureWebpack: {
    externals: {
      lodash: {
        commonjs: 'lodash',
        commonjs2: 'lodash',
        root: '_',
      },
    },
  },
}
```

Name of the package when used in a bundler

Name of the global variable when used in a browser

⚠️

Make sure to document this:

```
<script src="https://unkpg.com/lodash"></script>
<script src="https://unpkg.com/your-lib"></script>
```

When `your-lib` is used in a browser,
externalised dependencies have to be included before it

Handling CSS

/vue.config.js

```
module.exports = {
  lintOnSave: false,
  css: {
    extract: true,
  },
}
```

⚠️ **CSS has to be manually imported**

✅ **CSS can be omitted if people want to customise from scratch**

```
import Vue from 'vue'
import  YourLib from 'vue'
import 'your-lib/dist/YourLib.common.css'

Vue.use(YourLib)
```

```
module.exports = {
  lintOnSave: false,
  css: {
    extract: false,
  },
}
```

CSS bundled in JS

Injected into the DOM at runtime

⚠️ **You can't get rid of it**

✅ **No need to include a .css file**

Tree Shaking

What is a Tree Shaking ?

> *Tree shaking is a term commonly used in the JavaScript context for **dead-code elimination**.*
>
> *webpack comes with built-in support for **ES2015 modules** as well as unused module export detection. and [...] a „**sideEffects**" package.json property to denote which files in your project are "pure" and therefore safe to prune if unused.*

https://webpack.js.org/guides/tree-shaking/

This doesn’t work with bundled libraries!!

…but it works when the consumer imports our source!

- Webpack can’t really treeshake from already bundled code
- to profit from treeshaking, your *consumers* must include raw source
- There’s a couple of catches to this

/vue.config.js

Setting in the consumer's Vue CLI project

```
module.exports = {
  transpileDependencies: ["your-lib"],
}
```

- Will be transpired with the *consumer's* configuration
- Don't user any unusual config/plugins,
- If you must, then properly document it for your users.
- Don't use the de facto standard „@/path/alias“ for paths

https://cli.vuejs.org/config/#transpiledependencies

/package.json

```
{
  "name": „your-lib“,
  "sideEffects": false
}
```

⚠️

CSS from .vue files is a side-effect!

- Tells webpack your exports are side-effect free
- Allows more aggressive removal of unused exports
- Exceptions (files with side effects) can be configured

/package.json

```
{
  "name": „your-lib“,
  "sideEffects": ["*.css"]
}
```

✅

CSS marked as a side-effect!

- Tells webpack your exports are side-effect free
- Allows more aggressive removal of unused exports
- Exceptions (files with side effects) can be configured

The `--report` flag

```
$ vue-cli-service build --target lib --name VueFiledrop --dest dist/core src/index.js --report
```

- VueFiledrop.common-report.html
- VueFiledrop.common.js
- VueFiledrop.common.js.map
- VueFiledrop.umd-report.html
- VueFiledrop.umd.js
- VueFiledrop.umd.js.map
- VueFiledrop.umd.min-report.html
- VueFiledrop.umd.min.js
- VueFiledrop.umd.min.js.map

Datei | file:///Users/thorstenlunborg/Google%20Drive/Documents/Talks/2019-02-14%20Vuejs%20Amsterdam/assets/reporters/report-with-polyfills.html

VueFiledrop.common.js

node_modules

core-js

library

modules

es6.promise.js

_export.js

es6.array.iterator.js

web.dom.iterable.js

_array-includes.js

es7.promise.finally.js

_classof.js

_invoke.js

_to-primitive.js

_iter-detect.js

_string-at.js

_microtask.js

_object-dp.js

_new-promise-capability.js

es7.promise.try.js

_set-species.js

_shared.js

_object-keys-internal.js

_object-dps.js

_species-constructor.js

_global.js

...

_dom-create.js

es6.array.from.js

_iter-create.js

_iter-call.js

_iobject.js

_to-iobject.js

...

_redefine-all.js

_to-length.js

_is-array-iter.js

_an-instance.js

_object-keys.js

_ie8-dom-define.js

_descriptors.js

_iter-define.js

_object-create.js

es6.string.iterator.js

core.is-iterable.js

_hide.js

_shared-key.js

...

_defined.js

_uid.js

_to-integer.js

_ctx.js

_promise-resolve.js

_an-object.js

...

_perform.js

_html.js

_a-function.js

_set-to-string-tag.js

es6.array.is-array.js

_to-object.js

_core.js

_cof.js

_fails.js

_task.js

_for-of.js

_object-gpo.js

_wks.js

_create-property.js

_is-array.js

_user-agent.js

_has.js

_iter-step.js

...

_is-object.js

_redefine.js

_library.js

...

modules

es6.number.constructor.js

_iter-define.js

web.dom.iterable.js

_export.js

_array-includes.js

_set-proto.js

_string-trim.js

_to-primitive.js

_string-at.js

es6.string.iterator.js

_ctx.js

_object-gpo.js

_shared.js

_object-create.js

_object-dp.js

_object-dps.js

_inherit-if-required.js

_add-to-unscopables.js

_dom-create.js

_iobject.js

_wks.js

_hide.js

_to-length.js

_object-keys.js

_ie8-dom-define.js

_object-gopd.js

es6.array.iterator.js

es7.array.includes.js

_set-to-string-tag.js

_descriptors.js

_shared-key.js

_an-object.js

_object-keys-internal.js

_property-desc.js

_to-integer.js

...

_to-object.js

_object-gopn.js

es6.function.name.js

_string-ws.js

_html.js

_has.js

_is-object.js

_to-iobject.js

_redefine.js

_iter-create.js

_global.js

_defined.js

_a-function.js

_cof.js

_iter-step.js

...

_to-absolute-index.js

_uid.js

_core.js

_fails.js

_library.js

...

@vue/cli-service/lib/commands/build

entry-lib.js + 14 modules (concatenated)

vue-reactive-provide/dist

VueReactiveProvide.umd.min.js

...

array

from.js

...

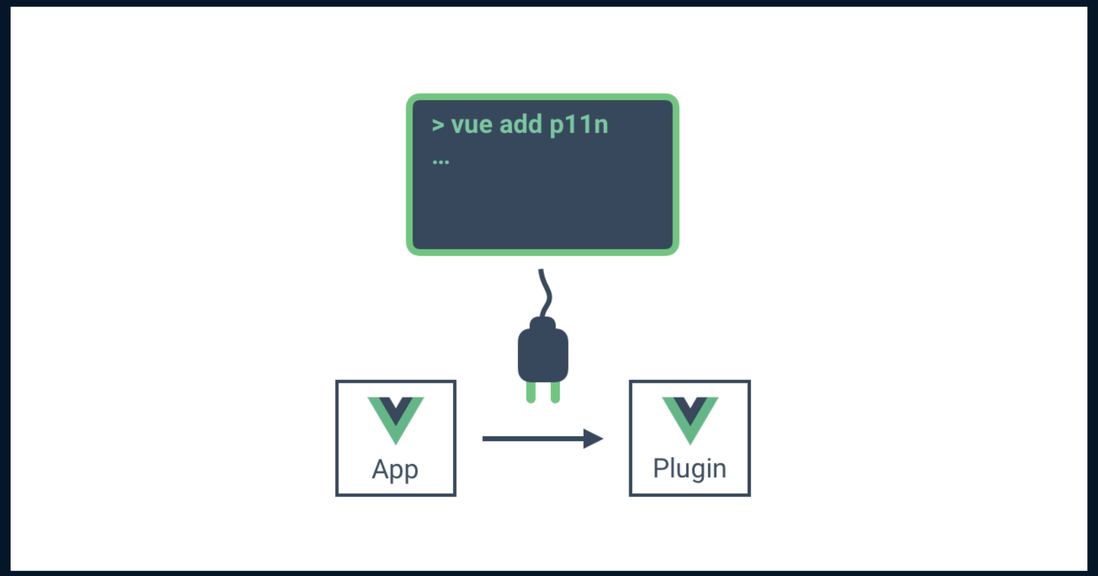
promise.js

...

Further Reading

You might also be interested in…

Github: kazupon



https://github.com/kazupon/vue-cli-plugin-p11n

https://medium.com/@kazu_pon/vue-cli-plugin-p11n-a51195ff7d3e

Github: chrisfritz

https://github.com/chrisvfritz/hello-vue-components

- Example repository with extensive comments
- lots of ticks and tricks
- build each component on it’s own
- … optionally into their own package(!!)

Github: linusborg
Twitter: @linus_borg