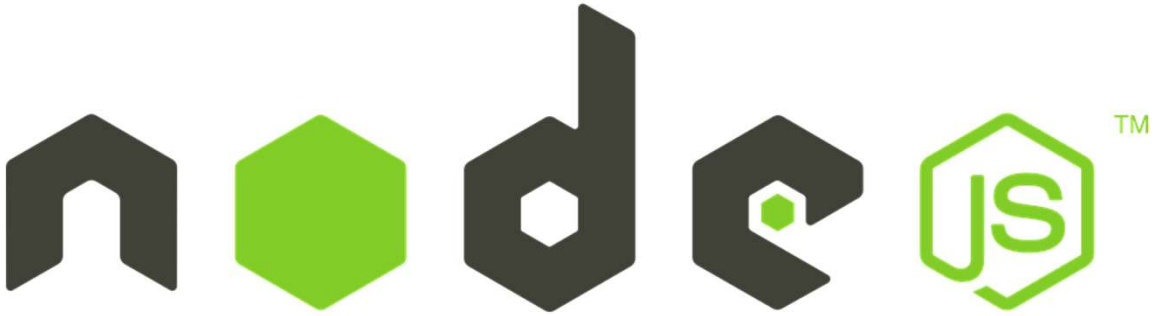


NPM – Publishing Vue Libraries

Peter Kassenaar – info@kassenaar.com

Creating Vue Libraries

Creating a reusable Vue Library

Let others ‘npm install’ your vue components

Problem

- You have a bunch of components, or other code that your organization needs to reuse

- You want your own theme, colours, look-and-feel and UI components to be used in your entire organization

Solution

- Publish the components on npm/nexus so anyone in your organization can do an `npm install` of the components.
- New versions can be fetched by using `npm update`.

- These slides are focused on Vue.js, but the same mechanism is true for other frameworks.
  - Look up the syntax difference yourself.

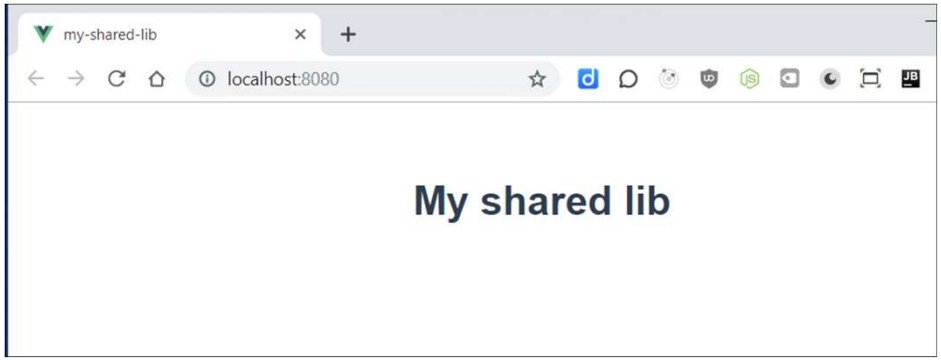
Step 1 – Create the Vue project

- Create the project that's going to contain the shared components, library code and so on.
- Using the Vue CLI
  - `vue create my-shared-lib`
  - Use CLI options as you like

```
Vue CLI v3.8.2
✨  Creating project in C:\Users\Gebruiker\Desktop\my-shared-lib.
🗃  Initializing git repository...
⚙  Installing CLI plugins. This might take a while...


> yorkie@2.0.0 install C:\Users\Gebruiker\Desktop\my-shared-lib\node
> node bin/install.js
```

- Update the default project per your needs
  - Remove `HelloWorld` component, update `App.vue` etc.
- `npm run serve`
  - Check if the application runs
  - We're not going to deploy an application, but it won't hurt to check your code/components



Step 2

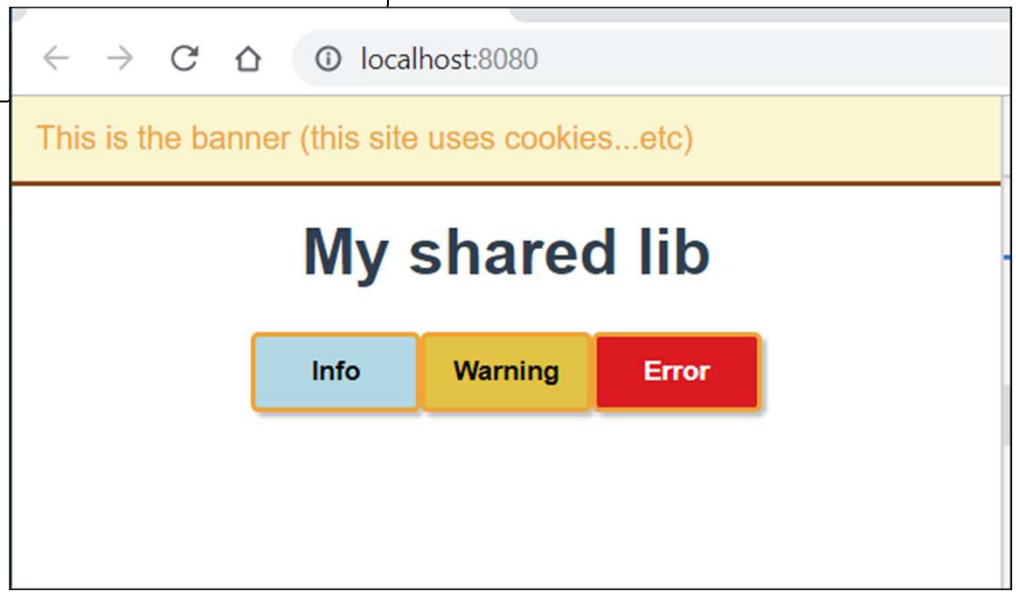
- Add/create your shared components/code
- We're going to create a configurable shared banner and a button
- See `./examples/200-vue-lib`

```
<template>
    <!--    Simple banner-->
    <div class="banner" :style="bannerStyles" :class="`banner__${position}`">
        <!--        Content projected here-->
        <slot></slot>
    </div>
</template>

<script>
    export default {
        name: "pk-banner",
   …
</script>
```

Example button + result

```
<template>
    <button @click="submit(type)"
            class="btn" :class="`btn__${type}`">
        <slot></slot>
    </button>
</template>
…
```



Step 3 – setting up library build

- Add a script to build a *library* instead of an application
- In `package.json` add
  - `--target lib` : create a library
  - `--name pklib` : name our library pklib
  - `./src/components/index.js` : entry point
- We're going to build this entry point!

```
"build-lib": "vue-cli-service build --target lib
              --name pklib ./src/components/index.js",
```

Step 4 Create the entry file `index.js`

```
// 1. import all the stuff we need
import Vue from 'vue';
import PkBanner from './pk-banner'
import PkButton from './pk-button'

// 2. create an object from our components
const components ={
    PkBanner,
    PkButton
};

// 3. iterate over the component object and
// add them to the global Vue instance
Object.keys(components).for(name => {
    Vue.component(name, components[name]);
});

// 4. export our components
export default components;
```

Step 5 – point towards output file

- We're going to build our lib momentarily.
- In `package.json`, point to the main output file in the generated `./dist` folder
- `CommonJS` bundle should be OK:
  - Vue-cli-service builds `commonJS` and `umd` bundles

```
"main": "./dist/pklib.common.js",
```

Step 6 – add `files` list to `package.json`

- Tell `npm` which files to upload.
- We're going to upload all files, so consumers also have access to the original `.vue` files if needed

```
"files": [
  "dist/*",
  "src/*",
  "public/*",
  "*.json",
  "*.js"
],
```

Step 7 – Verify npm user credentials

- Check if you are logged in to your npm account
- Create an accound and `adduser` if you haven't done that yet.

```
PS C:\Users\Gebruiker\Desktop\my-shared-lib> npm whoami
peter.kassenaar
PS C:\Users\Gebruiker\Desktop\my-shared-lib>
```

Step 8 – name your library

- Pick a name for your package, preferably using a scoped name.
- Make sure it’s not taken yet.
- Every name and version number has to be unique, even if you unpublished a package with that same name from npm before!

```
"name": "@peter.kassenaar/pk-vue-components",
```

Step 9 – build your library

- Build the bundles by using the script you added in Step 3
- `npm run build-lib`

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

 DONE  Compiled successfully in 4192ms

  File                      Size                     Gzipped

  dist\pklib.umd.min.js     18.70 KiB                6.24 KiB
  dist\pklib.umd.js         57.61 KiB                12.79 KiB
  dist\pklib.common.js      57.15 KiB                12.65 KiB
  dist\pklib.css            0.65 KiB                 0.30 KiB

  Images and other types of assets omitted.

PS C:\Users\Gebruiker\Desktop\my-shared-lib>
```

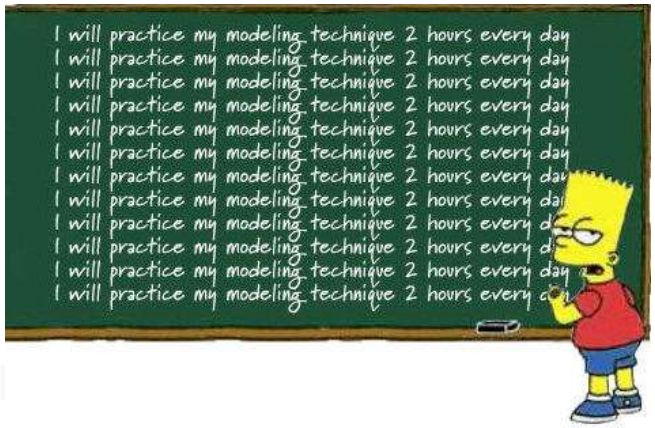
Step 10 – publish on npm!

- `npm publish --access=public`
- Set field `"private" : false` in `package.json` if npm is complaining about this.

```
npm notice 136B    src/main.js
npm notice === Tarball Details ===
npm notice name:          @peter.kassenaar/pk-vue-components
npm notice version:       0.1.0
npm notice package size:  201.6 kB
npm notice unpacked size: 803.9 kB
npm notice shasum:        68de3ff8c4452be07d3653e12de3ce17bf76ca71
npm notice integrity:     sha512-SponLfOd8+EOG[...]KmRNpalppUETg==
npm notice total files:   20
npm notice
+ @peter.kassenaar/pk-vue-components@0.1.0
PS C:\Users\Gebruiker\Desktop\my-shared-lib>
```

Workshop / Case

- Create your own Vue project and publish it on npm, using the steps described in this section
- Or: use `./examples/200-vue-project` as a starting point
  - Don't forget to update the `name`, `build` script, etc. in `package.json`!



Using your published library

- Create a new Vue project with the CLI
- Run an `npm install --save [libName]`
- `libName` is the name of your library, that you assigned in step 8.

```
PS C:\Users\Gebruiker\Desktop\use-vue-project> npm install "@peter.kassenaar/pk-vue-components"
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.9 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.9: wanted {"os":"d
} (current: {"os":"win32","arch":"x64"})

+ @peter.kassenaar/pk-vue-components@0.1.0
added 1 package and audited 24363 packages in 8.562s
found 0 vulnerabilities

PS C:\Users\Gebruiker\Desktop\use-vue-project>
```

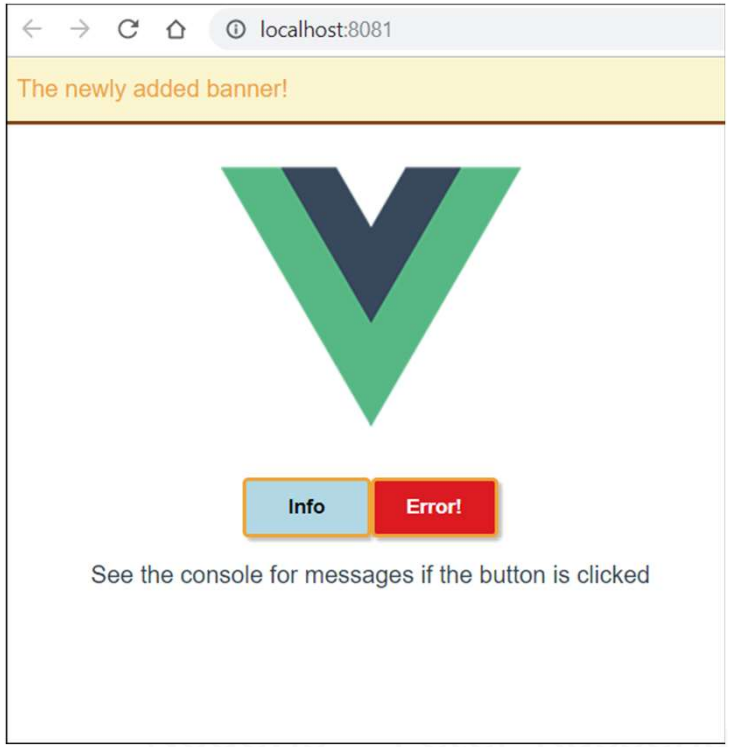
Updating main.js

- Import the library in `main.js` of the consuming application
- Also import the styles!
- B/c we added the components globally, we can use them directly in the app

```
// import our custom shared library
import '@peter.kassenaar/pk-vue-components';
import '@peter.kassenaar/pk-vue-components/dist/pklib.css’;
```

Using the components

```
<pk-banner>The newly added banner!</pk-banner>
<img alt="Vue logo" src="./assets/logo.png">
<pk-button>The info button</pk-button>
<pk-button type="error" @submit="onSubmit($event)">Error!</pk-button>
<p>See the console for messages if the button is clicked</p>
```



Maintaining your library

- Update components, code, CSS, as per usual
- Run `npm version major | minor | patch` to update the version number
- DON’T FORGET TO RUN THE BUILD SCRIPT BEFORE PUBLISHING!
  - `npm run build-lib`
- Run `npm publish` to publish the new library

Workshop / Case

- Use the library you created yourself in a new project
- Create a new project, import the correct files
- Update the original library (add a component, change some styles, etc)
- Update the version number
- Build the lib
- Publish the new version
- Update the consuming app.