Knowledge Friday

ANTON REINDL
18.03.2022

Pitch

Pinia

Pitch

Why


Evan You
@youyuxi
Replying to @olemarius @VueDose and @vuejs
Pinia is de facto Vuex 5! At this point it's really a naming/branding issue.
9:49 AM · Nov 24, 2021 · Twitter for iPhone

Pitch

LEARNING GOALS FOR TODAY

🎉 Get familiar with Pinia

What is Pinia at all?

💬 Get confident in using it

See an example of how to refactor from Vuex to

The Vue Store that you will enjoy using

💡 Intuitive

Stores are as familiar as components. API designed to let you write well organized stores.

🔑 Type Safe

Types are inferred, which means stores provide you with autocompletion even in JavaScript!

⚙️ Devtools support

Pinia hooks into Vue devtools to give you an enhanced development experience in both Vue 2 and Vue 3.

🔌 Extensible

React to store changes to extend Pinia with transactions, local storage synchronization, etc.

🏗 Modular by design

Build multiple stores and let your bundler code split them automatically.

📦 Extremely light

Pinia weighs around 1kb, you will forget it's even there!

Eduardo San Martin Marote

Author of Pinia, Vue Router & Vue.js Core Team Member

@posva

Pitch

VUEX

Pinia API is very different from Vuex ≤4, namely:

- mutations no longer exist. They were very often perceived as extremely verbose. They initially brought devtools integration but that is no longer an issue.
- No more nested structuring of modules. You can still nest stores implicitly by importing and using a store inside another but Pinia offers a flat structuring by design while still enabling ways of cross composition among stores. You can even have circular dependencies of stores.
- No namespaced modules. Given the flat architecture of stores, "namespacing" stores is inherent to how they are defined and you could say all stores are namespaced.
- No need to create custom complex wrappers to support TypeScript, everything is typed and the API is designed in a way to leverage TS type inference as much as possible.
- No more magic strings to inject. Import the functions, call them, enjoy autocompletion!
- No need to dynamically add stores, they are all dynamic by default and you won't even notice. Note you can still manually use a store to register it whenever you want but because it is automatic you don't need to worry about it.

Pitch

Pinia

```
npm install pinia @pinia/nuxt
```

NPM BUG

```
talks/nuxt-and-pinia  npm install pinia @pinia/nuxt
npm ERR! code ERESOLVE
npm ERR! ERESOLVE could not resolve
npm ERR!
npm ERR! While resolving: undefined@undefined
npm ERR! Found: vue@3.2.31
npm ERR! node_modules/vue
npm ERR!   peer vue@"3.2.31" from @nuxt/vite-builder@3.0.0-27458584.91fd16a
npm ERR!   node_modules/@nuxt/vite-builder
npm ERR!     @nuxt/vite-builder@"npm:@nuxt/vite-builder-edge@3.0.0-27458584.91fd16a" from nuxt3@3.0.0-27458584.91fd16a
npm ERR!     node_modules/nuxt3
npm ERR!       dev nuxt3@"latest" from the root project
npm ERR!   peer vue@"^3.2.25" from @vitejs/plugin-vue@2.2.4
npm ERR!   node_modules/@vitejs/plugin-vue
npm ERR!     @vitejs/plugin-vue@"^2.2.4" from @nuxt/vite-builder@3.0.0-27458584.91fd16a
npm ERR!     node_modules/@nuxt/vite-builder
npm ERR!       @nuxt/vite-builder@"npm:@nuxt/vite-builder-edge@3.0.0-27458584.91fd16a" from nuxt3@3.0.0-27458584.91fd16a
npm ERR!       node_modules/nuxt3
npm ERR!         dev nuxt3@"latest" from the root project
npm ERR!   4 more (@vue/server-renderer, @vueuse/head, nuxt3, vue-router)
npm ERR!
npm ERR! Could not resolve dependency:
npm ERR! @pinia/nuxt@"*" from the root project
npm ERR!
npm ERR! Conflicting peer dependency: vue@2.6.14
npm ERR! node_modules/vue
npm ERR!   peer vue@">= 2.5 < 3" from @vue/composition-api@1.4.9
npm ERR!   node_modules/@vue/composition-api
npm ERR!     peerOptional @vue/composition-api@"^1.4.0" from pinia@2.0.12
npm ERR!     node_modules/pinia
npm ERR!       pinia@"*" from the root project
npm ERR!       1 more (@pinia/nuxt)
npm ERR!
npm ERR! Fix the upstream dependency conflict, or retry
npm ERR! this command with --force, or --legacy-peer-deps
npm ERR! to accept an incorrect (and potentially broken) dependency resolution.
npm ERR!
npm ERR! See /Users/anton/.npm/eresolve-report.txt for a full report.

npm ERR! A complete log of this run can be found in:
npm ERR!     /Users/anton/.npm/_logs/2022-03-17T18_19_58_032Z-debug-0.log
✘  talks/nuxt-and-pinia
```

Pitch

Pinia

```
npm install --legacy-peer-deps pinia
```

Pinia

```
npm install pinia @pinia/nuxt
```

Pitch

MIGRATION FROM VUEX

Restructuring Modules to Stores

Pitch

Vuex

Vuex has the concept of a single store with multiple modules. These modules can optionally be namespaced and even nested within each other.

Pinia

The easiest way to transition that concept to be used with Pinia is that each module you used previously is now a store. Each store requires an id which is similar to a namespace in Vuex.

THIS CREATES A FLAT STRUCTURE FOR STORES

Pinia

```
# Vuex example (assuming namespaced modules)
src
└── store
    ├── index.js           # Initializes Vuex, imports modules
    └── modules
        ├── module1.js     # 'module1' namespace
        └── nested
            ├── index.js   # 'nested' namespace, imports module2 &
module3     ├── module2.js # 'nested/module2' namespace
            └── module3.js # 'nested/module3' namespace

# Pinia equivalent, note ids match previous namespaces
src
└── stores
    ├── index.js          # Optional
    ├── module1.js        # 'module1' id
    ├── nested-module2.js # 'nested/module2' id
    ├── nested-module3.js # 'nested/module3' id
    └── nested.js         # 'nested' id
```

Pitch

MIGRATION FROM VUEX

Actions

Pitch

ACTIONS

- can be called via components or in other actions
- can be called from other store actions
- are called directly on the store instance (no need for a "dispatch" method)
- can be asynchronous or synchronous
- can have as many parameters as desired
- can contain logic about how the state should be changed
- can change the state properties directly with this.propertyName or use a $patch method to group multiple state changes together for a single entry in the Vue devtools timeline

Pinia

```
actions:{
    async insertPost(post){
      // contains logic for altering differenct pieces of state
      try{
        await doAjaxRequest(post)

        // directly alter the state via the action and
    // change multiple pieces of state
        this.posts.push(post)
        this.user.postsCount++

    // OR alternatavley use .$patch to group change of posts and user.postsCount
devtools timeline
    this.$patch((state) => {
          state.posts.push(post)
          state.user.postsCount++
        })
      }catch(error){
        this.errors.push(error)
      }

    }
```

Pitch

3

MIGRATION FROM VUEX

MapHelpers

Pitch

HELPERS

mapState

There is only a mapState - it works for getters and state.

Because in Pinia that is equivalent.

Pinia

```
<template>
  <p>{{postsCount}} posts available</p>
</template>

<script>
import { mapState } from 'pinia'
import { usePostsStore } from "@/store/PostsStore";

export default {
  computed:{
    ...mapState(usePostsStore, ['postsCount'])
  }
};
</script>
```

Pitch

4

MIGRATION FROM VUEX

Converting a Module to Pinia

→ VSCODE

Pitch

What do you think?

Let's discuss

Pitch

Summary

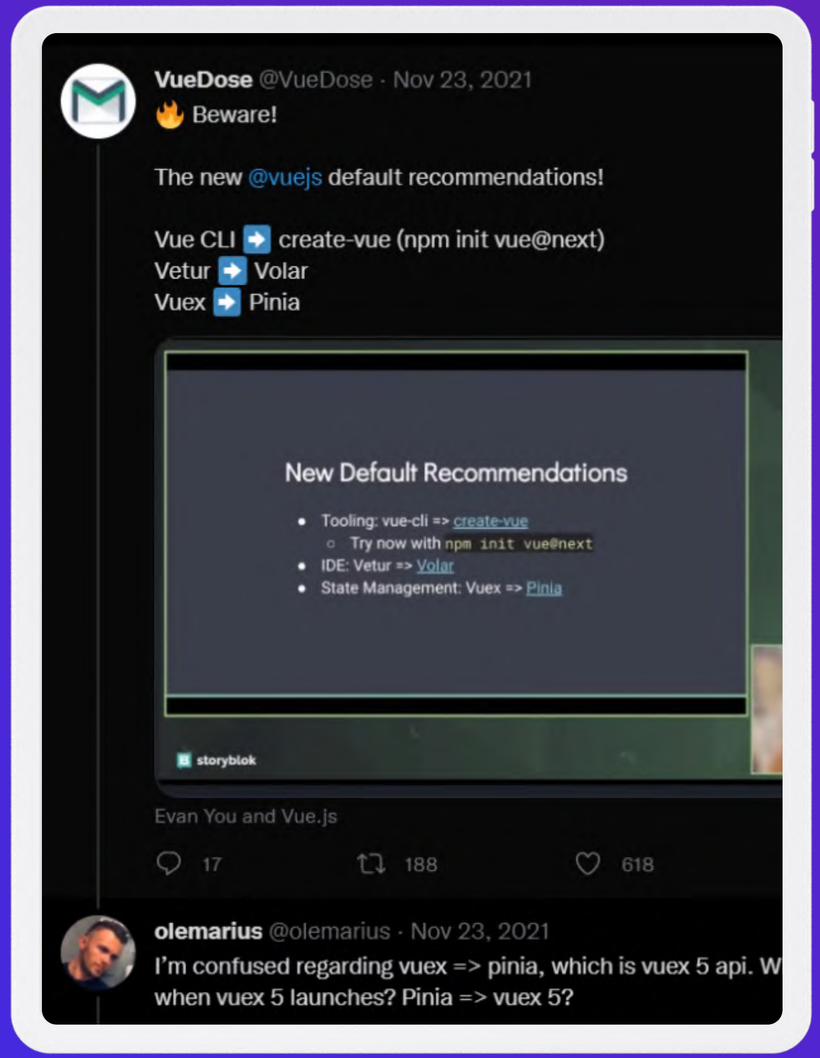
- Maintained by a Vue.js core team member and de-facto VueX 5
- Feels more like regular old javascript importing modules, calling actions as methods, accessing the state directly on the store, etc.
- Drops the need for mutations boilerplate
- Integrates with Vue Devtools and the Ecosystem

Pitch

VUE 3

Reommended Tooling:

- ☑ Vue CLI ⟶ create-vue
- ☑ IDE: Vetur ⟶ Vetur
- ☑ Vuex ⟶ Pinia



Pitch

TOPICS FOR NEXT TALKS

🎉 YARN, NPM OR PNPM

Battle of the package managers

💬 NUXT NUXI

The command line tool and why bother?

🍍 Pinia Plugins

Extend the basic functionality of Pinia

Pitch