What is "the gap"?


Vue 2
breaking changes
dependency conflicts
publishing strategies
Vue 3
Testing issues
code duplication

We have to...

1. Write compatible code that respects breaking changes
2. Write tests that can be run against both versions
3. Manage conflicting dependencies
4. Decide how to bundle and publish the project

Introducing Vue-Bridge

`@vue-bridge/runtime`

Polyfills some bridging code at runtime. Tiny (600b)

`@vue-bridge/testing`

Wrapper for Vue's test utils, providing unified API

`@vue-bridge/eslint-config`

Small set of helpful eslint rules

`@vue-bridge/vite-plugin`

Vite Plugin providing additional opitimizations at build time for edge cases

Documentation & Reference Material

Vue 2.7 will make this easier

- `Vue 2.7-alpha` was released this week
- Supports Composition API in Vue 2
- `vue-demi` / `@vue/composition-api` not required anymore
- allows optimization of a few things in VueBridge

It might be worth to wait until `2.7` drops

1. Writing compatible code

Example: v-model on Components

Vue 2

```
export default {
  props: {
    value: String
  },
  methods: {
    handleInput(e) {
      this.$emit('input', e.target.value)
    }
  }
}
```

Vue 3

```
export default {
  props: {
    modelValue: String
  },
  emits: ['update: modelValue'],
  methods: {
    handleInput(e) {
      this.$emit('update:modelValue', e.target.value)
    }
  }
}
```

Example: v-model - Manual Fix

```
export default {
  props: {
    modelValue: String
  },
  emits: ['update: modelValue'],
  model: {
    prop: 'modelValue',
    event: 'update:modelValue'
  },
  methods: {
    handleInput(e) {
      this.$emit('update:modelValue', e.target.value)
    }
  }
}
```

- Manual work every time
- Easy to miss
- Tricky with Typescript

Example: v-model - Vue-Bridge helper

```
import {defineComponent } from '@vue-bridge/runtime'

export default defineComponent({
  props: {
    modelValue: String
  },
  emits: ['update: modelValue'],
  methods: {
    handleInput(e) {
      this.$emit('update:modelValue', e.target.value)
    }
  }
})
```

- Write component for Vue 3 (future-proof)
- `@vue-bridge/runtime` adds model config
- works fine with TS

Unit testing against both Versions

Challenges for unit Testing

- Two versions of `@vue/test-utils` required (`v1` vs. `v2`)
- Subtle API differences between those versions

How to run the same tests against two different versions?

Example - Unit test

Vue 3 (test-utils `v2`)

```
import { mount } from '@vue/test-utils'

test('displays message', () => {
  const wrapper = mount(MessageComponent, {
    props: {
      msg: 'Hello world'
    },
    global: {
      provide: {
        store: mockStore
      }
    }
  })

  // Assert the rendered text of the component
  expect(wrapper.text()).toContain('Hello world')
})
```

Vue 2 (test-utils `v1`)

```
import { mount } from '@vue/test-utils'

test('displays message', () => {
  const wrapper = mount(MessageComponent, {
    propsData: {
      msg: 'Hello world'
    },
    provide: {
      store: mockStore
    }
  })

  // Assert the rendered text of the component
  expect(wrapper.text()).toContain('Hello world')
})
```

Example - Unit test

Using `@vue-bridge/testing`

```
- import { mount } from '@vue/test-utils'
+ import { mount } from '@vue-bridge/testing'

test('displays message', () => {
  const wrapper = mount(MessageComponent, {
    props: {
      msg: 'Hello world'
    },
    global: {
      provide: {
        store: mockStore
      }
    }
  })

  // Assert the rendered text of the component
  expect(wrapper.text()).toContain('Hello world')
})
```

- Write in `v2` style for Vue 3
- Options transformed for Vue 2
- TS Support
- More tricks in the docs later

Build-Time Optimizations

Build-time features

```
import { defineConfig } from 'vite'
import { createVuePlugin } from 'vite-plugin-vue2'
import { vueBridge } from '@vue-bridge/vite-plugin'

export default defineConfig({
  plugins: [
    createVuePlugin(),
    vueBridge({
      vueVersion: '2',
    }),
  ],
})
```

- Support for version-specific style blocks
- Support for version-specific imports
- alias trickery for easier source-sharing in monorepos

Version-specific styles

```
<template>
  <div class="root-wrapper">
    <slot />
  </div>
</template>
<style scoped v2>
  .root-wrapper /deep/ .class-in-slot {
    color: red;
  }
</style>
<style scoped v3>
  .root-wrapper  :deep(.class-in-slot) {
    color: red;
  }
</style>
```

- Different deep selectors
- Different transition styles
- escape hatch for various style fixes

Version-specific imports

```
import Teleport from './custom/teleport.ts?v-bridge'
```

```
// Compiled for Vue 3
import Teleport from './custom/teleport.vue3.ts'
```

```
// Compiled for Vue 2
import Teleport from './custom/teleport.vue2.ts'
```

- put version-specific code into dedicated `*.vueX.js` files

Managing conflicting dependencies

Dependency issues in flat repositories

```
{
  "name": "my-vue-package",
  "dependencies": {
    "lodash": "^4.17.21",
    "@vueuse/core": "^8.5.0"
  },
  "devDependencies": {
    "vue": "^3.2.32",
    "vue2": "npm:vue@^2.6.14",
    "@vitejs/plugin-vue": "^2",
    "vite-plugin-vue2": "^2",
    "vue-template-compiler": "^2.6.14",
  },
  "peerDependencies": {
    "vue": "^2.6 || ^3.2"
  }
}
```

Flat repositories are bad because:

- they require package aliases
- …which can lead to peer Dependency issues on install
- Packages relying on vue-demi make process cumbersome

A better solution: workspaces

Clean Dependencies with Workspaces (1)

```
# Folder Structure

lib-vue2/
├─ package.json
├─ vite.config.js
├─ src/                # --> Symlink to /lib-vue3/src

lib-vue3/
├─ package.json
├─ vite.config.js
├─ src/
  ├─ index.js
  ├─ MyComponent.vue

package.json
pnpm-workspaces.yaml
vite.config.shared.js   # shared build config
```

Clean Dependencies with Workspaces (2)

```
# Folder Structure

lib-vue2/
├ package.json
├ vite.config.js
├ src/                # --> Symlink to /lib-vue3/src

lib-vue3/
├ package.json
├ vite.config.js
├ src/
  ├ index.js
  ├ MyComponent.vue

package.json
pnpm-workspaces.yaml
vite.config.shared.js   # shared build config
```

Clean Dependencies with Workspaces (3)

```
# Folder Structure

lib-vue2/
├─ package.json
├─ vite.config.js
├─ src/                # --> Symlink to /lib-vue3/src

lib-vue3/
├─ package.json
├─ vite.config.js
├─ src/
  ├─ index.js
  ├─ MyComponent.vue

package.json
pnpm-workspaces.yaml
vite.config.shared.js   # shared build config
```

Bundling & Publishing

Bundling and Publishing as two Packages

```
// Vue 3
import { MyComponent } from 'vue3-my-package'
// Vue 2
import { MyComponent } from 'vue2-my-package'
```

```
lib-vue3
├─ dist/
  ├─ index.js # Vue 3 Bundle
├─ package.json
lib-vue2
  dist/
  ├─ index.js # Vue 2 Bundle
├─ package.json
```

```
{
  "name": "vue{2,3}-my-package"
  "exports": {
    ".": {
      "import": "./dist/index.js"
    },
  }
}
```

- Dedicated bundles for Vue 2 and Vue 3
- Separate packages for Vue 2 and Vue 3
- Clean separation of dependencies
- straightforward generation of type declarations

Bundling and Publishing as one Package

```
// Vue 3
import { MyComponent } from 'vue-my-package'
// Vue 2
import { MyComponent } from 'vue-my-package/vue2'
```

```
lib-vue3
  ├ dist/
    ├ index.js # Vue 3 Bundle
  dist-vue2/
    ├ index.js # Vue 2 Bundle
  package.json
```

```
{
  "name": "vue-my-package",
  "exports": {
    ".": {
      "import": "./dist/index.js"
    },
    "./vue2": "./dist-vue2/index.js"
  }
}
```

- Dedicated bundles for Vue 2 and Vue 3
- One package to install regardless of target version
- Possibly more conflicts with `peerDependencies`
- Trickier to get right with type declarations

Demo Time!

https://github.com/vue-bridge/template-monorepo

Vue-Bridge

Getting Started Topics How-to Guides Reference

VueBridge

Tooling & Templates for

Vue-Bridge needs you!

- Add missing Documentation chapters
- Review & proof-read existing Documentation
- Provide additional Templates
- Contribute eslint rules
- Write proper end-to-end tests
- … and so much more!

Become a contributor!

Docs	https://vue-bridge.docs.netlify.app
Repo	https://github.com/vue-bridge/vue-bridge
Template	https://github.com/vue-bridge/template-monorepo
Twitter	https://twitter.com/VueBridge

You made it! We're done!

Questions?

This talk was built with Slidev

https://sli.dev

@Linus_Borg

github.com/linusborg

No bundling - Publishing raw SFCs

```
// Vue 3
import { MyComponent } from 'vue-my-package'
// Vue 2
import { MyComponent } from 'vue-my-package'
```

```
# Folder Structure
src/
├─ index.js
├─ MyComponent.vue
```

`package.json`

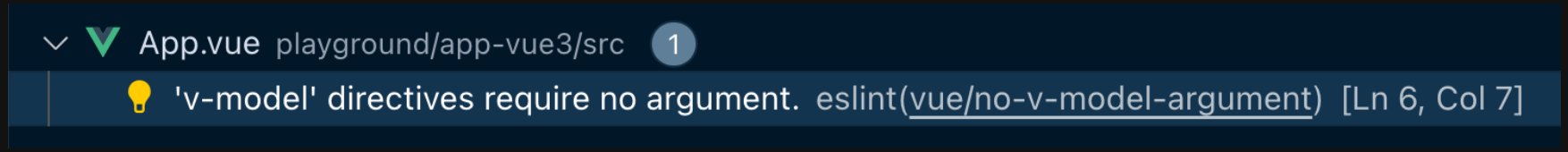
```
{
  "main": "src/index.js"
}
```

- No build test necessary (for JS)
- Can be published in one package
- Adds SFC compilation overload to consuming app
- Needs additional tooling when using TS
- Unit tests for both Versions need trickery

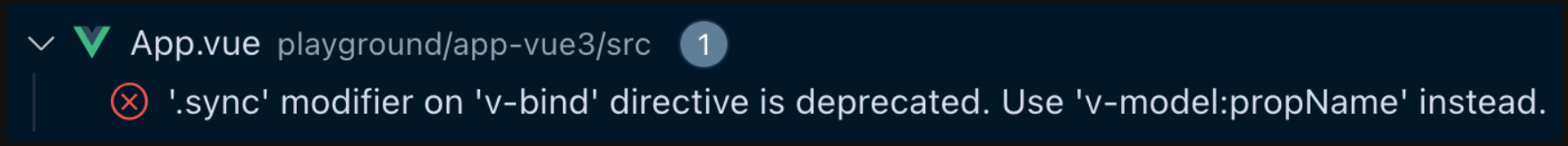
Three Levels of support

1. Have them handled for you
2. Have them be pointed out to you
3. Have resources to consult about them.

Linting for compat issues



vs:



`@vue-bridge/eslint-config` preset uses rules from `eslint-plugin-vue` for both versions: