浅谈Vue3

-- johnsqliu

目录

About Vue3 Version

Vue 3 is now the new default version!

About Vue3 Version

Vue 3 is now the new default version!

Can I USE Vue 3 ?

About Vue3 Version

Vue 3 is now the new default version!

the official release of Vue.js 3.0 "One Piece".

The Vue Point

GitHub Source · RSS Feed · Vuejs.org →

September 18, 2020

Announcing Vue 3.0 "One Piece"

Evan You
@youyuxi

NEXT ARTICLE
Reflections for 2020-2021

← Back to the blog

Today we are proud to announce the official release of Vue.js 3.0 "One Piece". This new major version of the framework provides improved performance, smaller bundle sizes, better TypeScript integration, new APIs for tackling large scale use cases, and a solid foundation for long-term future iterations of the framework.

Vue 3 as the New Default

The Vue Point

GitHub Source · RSS Feed · Vuejs.org →

January 20, 2022

Vue 3 as the New Default

Evan You
@youyuxi

PREVIOUS ARTICLE
Vue 3.2 Released!

← Back to the blog

TL;DR: Vue 3 is now the new default version as of **Monday, February 7, 2022**!

Make sure to read the Potential Required Actions section to see if you need to make certain changes before the switch to avoid breakage.

From a Library to a Framework

When Vue first started, it was just a runtime library. Over the years, it has evolved into a framework that encompasses many sub projects:

- The core library, i.e. the `vue` npm package

About Vue3 Version

Vue 3 is now the new default version!

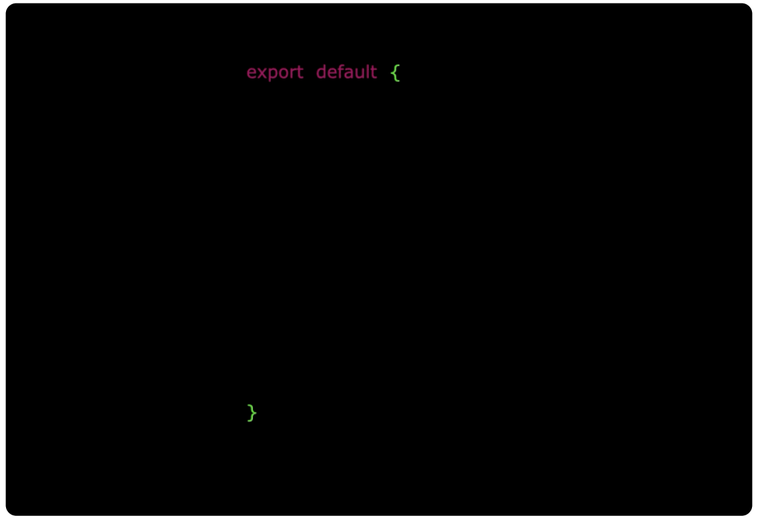
主要改进

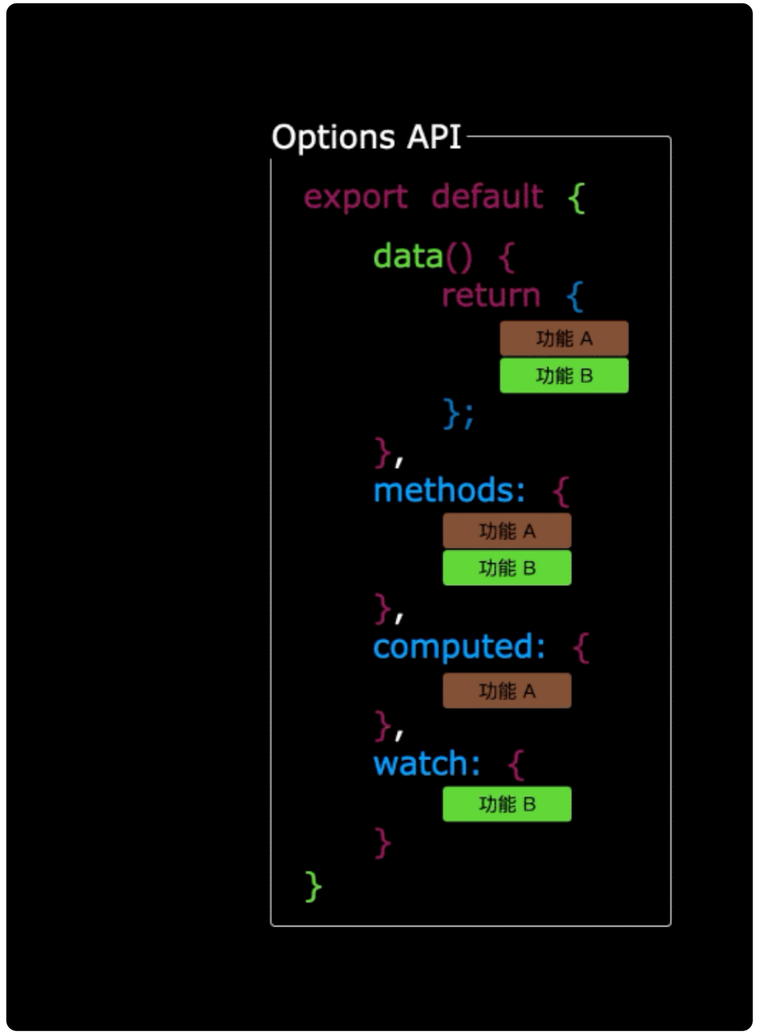
- 有重大改进
- 合理的实现 & 维护成本
- 兼容性90%的Api

升级的挑战在于：兼容性&依赖内部的一些行为

Composition API

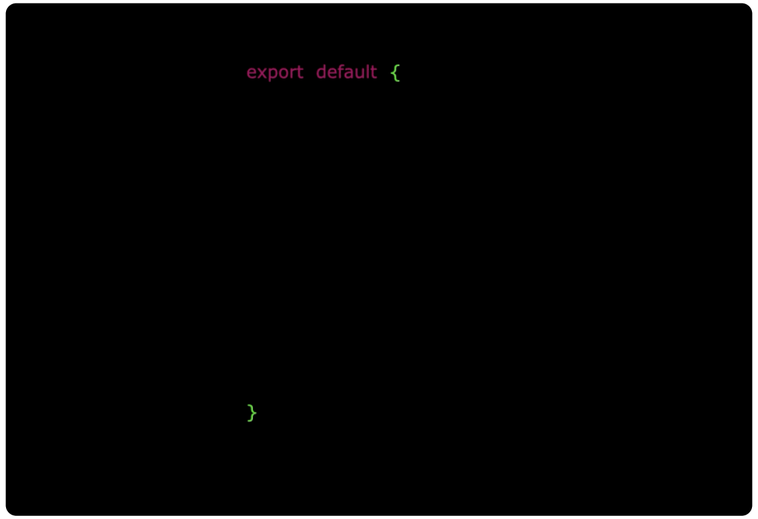
```
export default {
  data() {
    return {
      value: '',
    }
  },
  computed() {
    val2() {
      return `hi ${this.value}`;
    },
  },
  watch() {
    value(newVal, oldVal) {
      this.val2 = `hi ${this.value}`;
    },
  },
  methods: {
    fn1() {},
    fn2() {},
  }
}
```

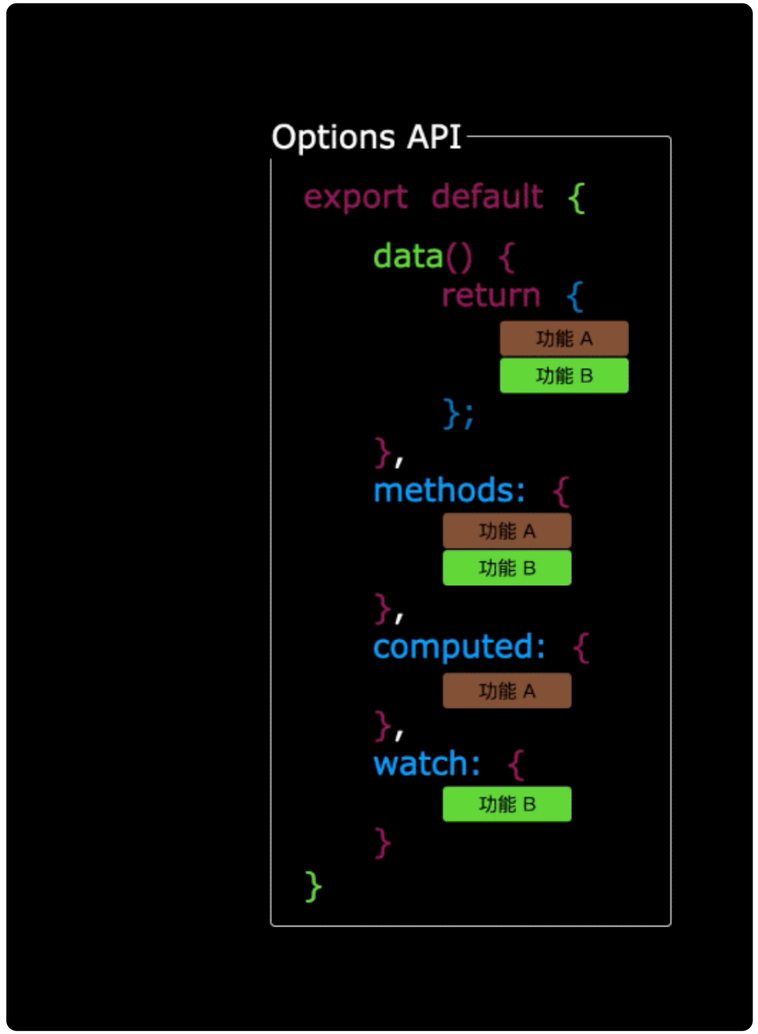




Composition API

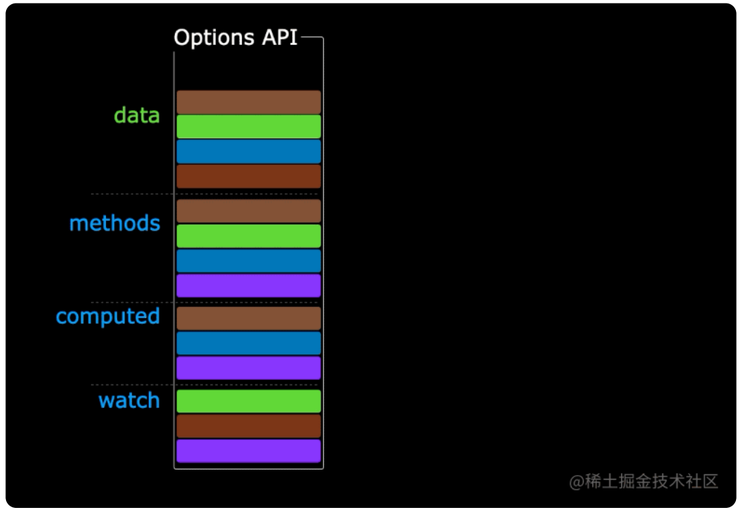
```
export default {
  data() {
    return {
      value: '',
    }
  },
  computed() {
    val2() {
      return `hi ${this.value}`;
    },
  },
  watch() {
    value(newVal, oldVal) {
      this.val2 = `hi ${this.value}`;
    },
  },
  methods: {
    fn1() {},
    fn2() {},
  }
}
```

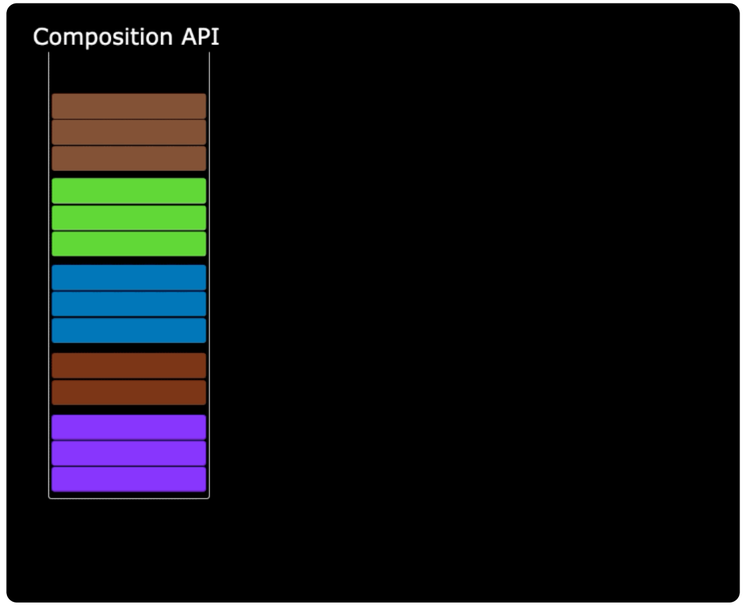




Composition API

```
const fn = async ()=> {
  //const { data } = await fetch(..)
}
const fn2 = ()=> { ... }
// page1
export default defineComponent({
  setup() {
    const value = ref(2);
    const val2 = computed(() => 2 * va
    fn2()
    return {
      value, val2,
      fn,
      fn2
    }
  }
});
// page2
export default defineComponent({
  setup() {
    return {
```





Vue 3 New Feature

Vue 3 New Feature

setup

props

```
// MyBook.vue

import { toRefs } from 'vue'

setup(props) {
  const { title } = toRefs(props)

  console.log(title.value)
}
```

> 但是，因为 props 是响应式的，你不能使用 ES6 解构，它会消除 prop 的响应性。

context

```
// MyBook.vue
export default {
  setup(props, { attrs, slots, emit, expose }) {
    ...
  }
}
```

> 在 setup() 内部，this 不是该活跃实例的引用，因为 setup() 是在解析其它组件选项之前被调用的，所以 setup() 内部的 this 的行为与其它选项中的 this 完全不同。这使得 setup() 在和其它选项式 API 一起使用时可能会导致混淆。

reactivity

Vue 3 New Feature

`Lifecycle Hooks`

你可以通过在生命周期钩子前面加上 “on” 来访问组件的生命周期钩子。在 `setup ()` 内部调用生命周期钩子：

选项式 API	Hook inside setup	选项式 API	Hook inside setup
`beforeCreate`	`Not needed*`	`beforeUnmount`	`onBeforeUnmount`
`created`	`Not needed*`	`unmounted`	`onUnmounted`
`beforeMount`	`onBeforeMount`	`errorCaptured`	`onErrorCaptured`
`mounted`	`onMounted`	`renderTracked`	`onRenderTracked`
`beforeUpdate`	`onBeforeUpdate`	`renderTriggered`	`onRenderTriggered`
`updated`	`onUpdated`	`activated`	`onActivated`
		`deactivated`	`onDeactivated`

reactivity

Vue 3 New Feature

Reactivity

`vue3`使用 `Proxy` 代替`vue2`的 `Object.defineProperty()` 作为其响应性的核心。

```
function ref(value) {
  const refObject = {
    get value() {
      track(refObject, 'value')
      return value
    },
    set value(newValue) {
      trigger(refObject, 'value')
      value = newValue
    }
  }
  return refObject
}
```

```
function reactive(obj) {
  return new Proxy(obj, {
    get(target, key) {
      track(target, key)
      return target[key]
    },
    set(target, key, value) {
      trigger(target, key)
      target[key] = value
    }
  })
}
```

> 关于更多vue3响应式对象，包括 `ref` 、 `reactive` 、 `watch` 、 `watchEffect` 、 `computed` 、 `toRef` 、 `toRefs` ，详细研究官方文档

reactivity

TypeScript integration

Vue 3's codebase is written in TypeScript, with automatically generated, tested, and bundled type definitions so they are always up-to-date. Composition API works great with type inference.

TypeScript integration

Vue 3's codebase is written in TypeScript, with automatically generated, tested, and bundled type definitions so they are always up-to-date. Composition API works great with type inference.

项目配置

```
// tsconfig.json
{
  "compilerOptions": {
    "target": "esnext",
    "module": "esnext",
    // 这样就可以对 `this` 上的数据属性进行更严格的推断
    "strict": true,
    "jsx": "preserve",
    "moduleResolution": "node"
  }
}
```

Webpack 配置

```
// webpack.config.js
module.exports = {
...
module: {
  rules: [
  {
    test: /\.tsx?$/,
    loader: 'ts-loader',
    options: {
      appendTsSuffixTo: [/\.vue$/],
    },
    exclude: /node_modules/,
  },
  {
    test: /\.vue$/,
    loader: 'vue-loader',
  }
  ...
```

TypeScript integration

Vue 3's codebase is written in TypeScript, with automatically generated, tested, and bundled type definitions so they are always up-to-date. Composition API works great with type inference.

项目构建

使用 `Vue CLI` 可以生成使用 `TypeScript` 的新项目，或者用 `vite` 生成使用 `TypeScript` 的新项目

但需要确保组件的 `script` 部分已将语言设置为 `TypeScript`

```
<script lang="ts">
  ...
</script>
```

编辑器支持

对于使用 TypeScript 开发 Vue 应用程序，我们强烈建议使用 Visual Studio Code，它为 TypeScript 提供了很好的开箱即用支持。如果你使用的是单文件组件 (SFCs)，那么可以使用很棒的 `Volar extension` ，它在 SFCs 中提供了 TypeScript 推理和许多其他优秀的特性。 WebStorm 同时为 TypeScript 和 Vue 提供内置的支持。其它的 JetBrains IDE 对它们也通过内置或免费插件的方式进行支持。

TypeScript integration

Vue 3's codebase is written in TypeScript, with automatically generated, tested, and bundled type definitions so they are always up-to-date. Composition API works great with type inference.

定义 Vue 组件

要让 TypeScript 正确推断 Vue 组件选项中的类型，需要使用 defineComponent 全局方法定义组件：

```
import { defineComponent } from 'vue'

const Component = defineComponent({
  // 已启用类型推断
})
```

如果你使用的是单文件组件，则通常会被写成：

```
<script lang="ts">
import { defineComponent } from 'vue'
export default defineComponent({
  // 已启用类型推断
})
</script>
```

vue3配套

vue3配套

vue-router

配合 `Vue 3` 官方也提供了 `Vue router 4.x` 版本的升级。

> 因为我们在 setup 里面没有访问 this，所以我们不能再直接访问 this.*router*或*this*.route。作为替代，我们使用 useRouter 函数：

```
import { useRouter, useRoute } from 'vue-router'

export default {
  setup() {
    const router = useRouter()
    const route = useRoute()

    function pushWithQuery(query) {
      router.push({
        name: 'search',
        query: {
          ...route.query,
        },
      })
    }
  },
}
```

vue3配套

状态管理

Vuex 4.x

`Vuex 4.x` 是与 `Vue 3` 配套的官方提供的状态管理 + 库。 在 Composition Api模式下，可以通过调用 `useStore` 函数，来在 `setup` 钩子函数中访问 `store` 。这与在组件中使用选项式 API 访问 this.$store 是等效的。

```
import { useStore } from 'vuex'

export default {
  setup () {
    const store = useStore()
  }
}
```

Pinia

Pinia 是 Vue.js 的轻量级状态管理库，最近很受欢迎。它使用 Vue 3 中的新反应系统来构建一个直观且完全类型化的状态管理库。

pinia会替代vuex吗

很大概率，都是core team成员，讨论结果未来形态会很像pinia，新东西都写在pinia里，如果开发一个新项目，如果使用ts，推荐使用pinia。

```
// stores/counter.js
import { defineStore } from 'pinia'

export const useCounterStore = defineStore('counter', {
state: () => {
  return { count: 0 }
},
// could also be defined as
// state: () => ({ count: 0 })
actions: {

  increment() {
```

vue3配套

VueUse

针对vue3 composition api的工具库

> VueUse is a collection of utility functions based on Composition API. We assume you are already familiar with the basic ideas of Composition API before you continue.

0

Mouse X

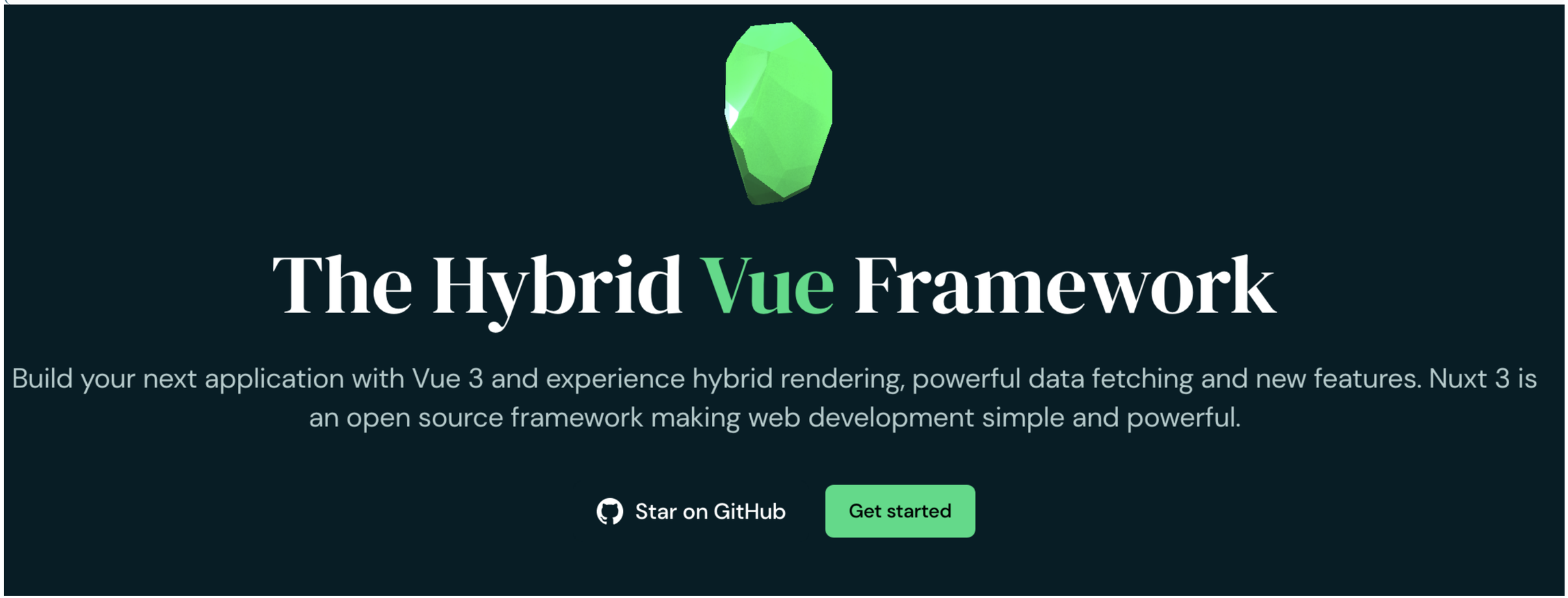
0

Mouse Y

vue3配套

Nuxt 3

> Build your next application with Vue 3 and experience hybrid rendering, powerful data fetching and new features. Nuxt 3 is an open source framework making web development simple and powerful.



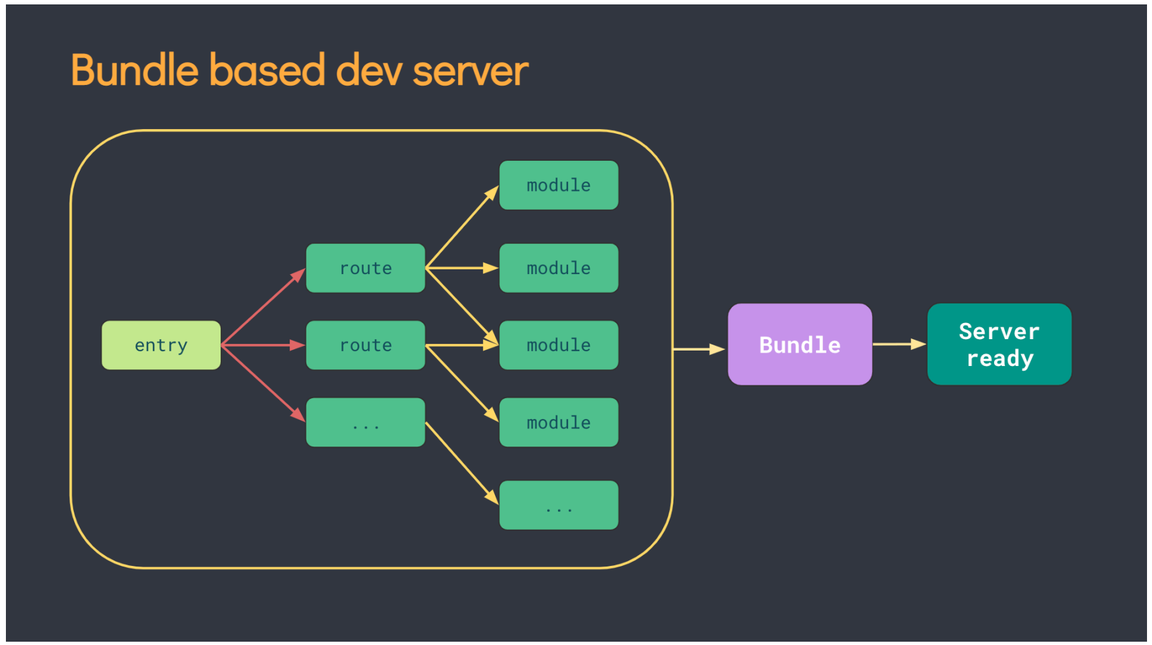
vue3配套

UI框架

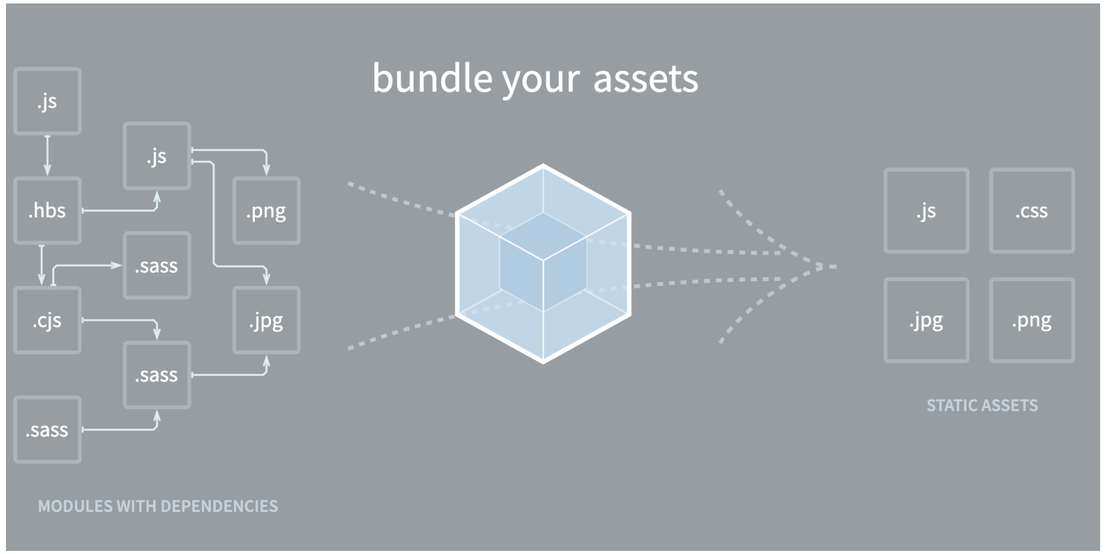
- 目前兼容的UI框架：Quasar、Vuetify(alpha)、NaiveUI、Prime Vue、Element Plus、Ant Design Vue
- 移动端组件库：Ionic Vue、Vant、Varlet
- tdesign vue-next

vite

> Vite 旨在利用生态系统中的新进展解决上述问题：浏览器开始原生支持 ES 模块，且越来越多 JavaScript 工具使用编译型语言编写。



webpack



从上图我们可以看出来，Webpack Dev Server 在启动时，需要先打包一遍，然后启动开发服务器，这一过程是需要耗费很多时间的。而vite是直接启动Server，并不会先编译所有的代码文件

在进行热更新时，Webpack 修改某个文件过后，会自动以这个文件为入口重写 build一次，所有的涉及到的依赖也都会被加载一遍，所以反应速度会慢很多。而Vite 只需要立即编译当前所修改的文件即可，所以 响应速度非常快

Webpack 工具的做法是将所有模块提前编译，不管模块是否会被执行，都要被编译和打包到 bundle 里。随着项目越来越大打包后的

Others

vue2 兼容升级

- **vueuse**： 🎩 From v4.0, it works for Vue 2 & 3 within a single package by the power of vue-demi!
- **@vue/composition-api**
- **vue 2.7**： 披着vue3外壳的vue2

部署依赖包兼容

- nodejs版本要求: Vite 需要 Node.js 版本 >= 12.0.0
- 、、、

Thank You!