# vue3

## vue2和vue3的区别

### 生命周期

|  |  |
| --- | --- |
| vue2 | vue3 |
| beforeCreate/created | setup |
| beforeMount | onBeforeMount |
| mounted | onMounted |
| beforeUpdate | onBeforeUpdate |
| updated | onUpdated |
| beforeUnmount | onBeforeUnmount |
| unmounted | onUnmounted |

### 多根节点

Vue2只支持单个根节点

Vue3 支持多个根节点，也就是 fragment。即以下多根节点的写法是被允许的。

到一起，增强了代码的可读性、内聚性，其还提供了较为完美的逻辑复用性方案。

### 组合式API和选项式API

|  |  |  |  |
| --- | --- | --- | --- |
| vue | 结构 | 特点 |  |
| 组合式API | vue3 | 将一个功能的所有代码集中起来处理 | 对复杂组件更易于维护和理解 |
| 选项式API | vue2 | 使用特定的选项对象（如 data、methods、computed、watch 等）来组织组件逻辑，每个选项负责一种特定的功能 | 对于简单组件，易于理解和使用；但组件变得复杂时，相关逻辑分散在不同的选项中，难以维护 |

### 异步组件（Suspense）

Vue3 提供 Suspense 组件，允许程序在等待异步组件加载完成前渲染兜底的内容，如 loading ，使用户的体验更平滑。使用它，需在模板中声明，并包括两个命名插槽：default 和 fallback。Suspense 确保加载完异步内容时显示默认插槽，并将 fallback 插槽用作加载状态。

### Teleport

Vue3 提供 Teleport 组件可将部分 DOM 移动到 Vue app 之外的位置。比如项目中常见的 Dialog 弹窗。

### 响应式原理

|  |  |  |
| --- | --- | --- |
| 特征 | Vue 2 | Vue 3 |
| 响应式实现 | 使用 Object.defineProperty方法来定义对象的 getter 和 setter，以实现数据的响应式；  当数据被访问时，getter 被调用，Vue 会记录依赖（即哪些组件使用了这个数据）；  当数据被修改时，setter 被调用，Vue 会通知所有依赖该数据的组件进行更新 | 使用 Proxy，允许对整个对象进行代理，而不是逐个属性定义 getter 和 setter。  Proxy 可以拦截对象的各种操作（如属性读取、修改、添加、删除等），从而实现更全面的响应式。 |
| 新属性支持 | 不支持，需要使用 Vue.set | 支持，直接添加新属性自动响应 |
| 数组索引支持 | 不支持，需使用 splice 或 Vue.set | 支持，数组操作均可触发更新 |
| 性能 | 性能开销较大，尤其在大规模应用中 | 性能更佳，响应式开销更低 |
| 依赖追踪 | 依赖追踪机制较简单 | 更复杂的依赖追踪机制 |
| 复杂性 | 简单但功能有限 | 更复杂，但更灵活和强大 |

### 虚拟DOM

Vue3 相比于 Vue2，虚拟DOM上增加 patchFlag 字段。我们借助Vue3 Template Explorer来看。

### 事件缓存

Vue3 的cacheHandler可在第一次渲染后缓存我们的事件。相比于 Vue2 无需每次渲染都传递一个新函数。加一个 click 事件。

### Diff算法优化

### 打包优化

Tree-shaking：模块打包 webpack、rollup 等中的概念。移除 JavaScript 上下文中未引用的代码。主要依赖于 import 和 export 语句，用来检测代码模块是否被导出、导入，且被 JavaScript 文件使用。

以 nextTick 为例子，在 Vue2 中，全局API暴露在Vue实例上，即使未使用，也无法通过 tree-shaking 进行消除。

Vue3 中针对全局和内部的API进行了重构，并考虑到 tree-shaking 的支持。因此，全局API现在只能作为ES模块构建的命名导出进行访问。

通过这一更改，只要模块绑定器支持 tree-shaking，则Vue应用程序中未使用的 api 将从最终的捆绑包中消除，获得最佳文件大小。

受此更改影响的全局API如下所示。

Vue.nextTick

Vue.observable （用 Vue.reactive 替换）

Vue.version

Vue.compile （仅全构建）

Vue.set （仅兼容构建）

Vue.delete （仅兼容构建）

内部API也有诸如 transition、v-model 等标签或者指令被命名导出。只有在程序真正使用才会被捆绑打包。Vue3 将所有运行功能打包也只有约22.5kb，比 Vue2 轻量很多。

### TypeScript支持

Vue3 由 TypeScript 重写，相对于 Vue2 有更好的 TypeScript 支持。

Vue2 Options API 中 option 是个简单对象，而 TypeScript 是一种类型系统，面向对象的语法，不是特别匹配。

Vue2 需要vue-class-component强化vue原生组件，也需要vue-property-decorator增加更多结合Vue特性的装饰器，写法比较繁琐。

### 状态管理

vue2：使用Vuex，通过 state、getters、mutations 和 actions 组织，采用单一的全局状态树，所有状态集中在一个地方，便于管理和调试

vue3：使用Pinia,允许多个 store 模块，支持模块化的状态管理，更加灵活

## 1. 什么是Vue3？ Vue3有哪些新增特性？

答：Vue3是Vue.js框架的最新版本，它增加了很多新特性，包括Composition API、Teleport、Suspense 和Fragment等。

## 2. Vue3 Composition API是什么？它的作用是什么？

答：Vue3 Composition API是Vue3中的一个新特性，它的作用是将组件中的逻辑分解成可复用的可组合函数。通过使用Composition API，可以更好地组织代码和管理状态。

## 3. Vue3中的Teleport是什么？它的作用是什么？

答：Vue3中的Teleport 是控制渲染位置的新指令。它的作用是在DOM中移动一个组件的内容而不改变组件的父级。

## 4. Vue3中的Suspense是什么？它的作用是什么？

答：Vue3中的Suspense是Vue3中新增的一个组件，它的作用是实现延迟加载和错误处理。在组件中加入Suspense，可以让异步组件可以渲染出加载状态，并且如果异步组件加载时出现错误，也能够处理这些错误。

## 5. Vue3中的Fragment是什么？它的作用是什么？

答：Vue3中的Fragment是用来承载多个子元素的虚拟组件。它的作用是可以解决在Vue2中，使用v-for迭代元素时需要添加一个包装元素的问题。

## 6. 什么是响应式系统？ Vue3中的响应式系统有哪些更新？

答：响应式系统是Vue中的核心概念之一，它允许在状态发生变化时更新视图。Vue3中的响应式系统更新包括Proxy、Reflect和WeakMap等。

## 7. Vue3中的事件修饰符有哪些？

答：Vue3中的事件修饰符与Vue2基本相同，包括stop、prevent、capture和self等。

## 8. Vue3中的指令有哪些？

答：Vue3中的指令包括v-if、v-for、v-bind、v-on、v-html、v-model、v-show、v-slot、v-text等。

## 9. Vue3中如何实现动态组件？

答：Vue3中使用 <component> 元素和 v-bind:is 属性来实现动态组件。例如， <component v-bind:is="currentComponent"></component> 。

## 10. Vue3如何实现异步组件加载？

答：Vue3中使用 import() 来异步加载组件。

## 11. Vue3如何实现插槽？

答：Vue3中使用 <slot name="slot-name"></slot> 来实现插槽。在父组件中使用 <template v-slot:slot-name></template> 来填充插槽。

## 12. Vue3如何实现自定义指令？

答：Vue3使用 app.directive() 方法来注册指令，例如 app.directive('focus', {mounted(el) {el.focus()}}) 。

## 13. Vue3如何实现混入？

答：Vue3使用 app.mixin() 方法来注册混入，例如 app.mixin({created() {console.log('mixin created')}}) 。

## 14. Vue3如何实现自定义渲染函数？

答：Vue3使用 h() 函数来创建虚拟节点，例如 h('div', {class: 'container'}, 'Hello, world') 。

## 15. Vue3中的响应式系统如何处理循环引用问题？

答：Vue3中使用WeakMap来处理循环引用问题。

## 16. Vue3如何实现全局状态管理？

答：Vue3中使用 provide() 和 inject() 函数来实现全局状态管理。

## 17. Vue3中的ref指令有哪些用途？

答：Vue3中的ref指令可以用来在组件内部获取子组件的实例，也可以用来获取DOM元素或其他组件的实例。

## 18. Vue3中的setup()函数有什么用途？

答：Vue3中的setup()函数是用来替代Vue2中的data、methods和computed等选项的。它可以用来创建响应式数据和添加需要在模板中使用的方法。

## 19. Vue3如何使用provide和inject实现依赖注入？

答：在父组件中使用 provide() ，并在子组件中使用 inject() 来注入依赖项。

## 20. Vue3如何实现异步验证表单输入？

答：使用 watch() 函数，监听表单输入的变化，并使用异步函数处理验证逻辑。

## 21. Vue3中如何使用路由？

答：Vue3中使用Vue Router来实现路由。首先需要安装Vue Router，然后使用 createRouter() 函数创建路由对象，然后在根Vue实例中使用 app.use() 方法注册Vue Router。

## 22. Vue3中的provide注入的依赖项如何在子组件中更新？

答：通过给provide注入的对象添加响应式属性来让子组件能够更新依赖项。

## 23. Vue3中如何使用axios发送HTTP请求？

答：在Vue3中使用axios发送HTTP请求，需要先安装axios，并在组件中导入axios。然后可以使用axios的get、post、put、delete等方法来发送HTTP请求。

## 24. Vue3如何使用vuex进行状态管理？

答：Vue3中使用Vuex进行状态管理，需要先安装Vuex，并在根Vue实例中使用 app.use() 方法注册Vuex。然后在组件中使用 store 选项来创建和访问Vuex的状态。

## 25. Vue3中如何使用emit事件来与父组件通信？

答：在子组件中使用 this.$emit() 方法触发 emit 事件，并将需要传递的数据作为参数传递给父组件。

## 26. Vue3中如何使用slot来构建可复用组件？

答：在组件中使用 <slot> 元素来定义插槽，在父组件中使用 <template v-slot:slot-name> 来填充插槽。

## 27. Vue3中如何处理条件渲染？

答：使用 v-if 指令来实现条件渲染。

## 28. Vue3中如何处理列表渲染？

答：使用 v-for 指令来实现列表渲染。

## 29. Vue3中如何处理动态绑定属性？

答：使用 v-bind 指令来实现动态绑定属性。

## 30. Vue3中如何处理事件绑定？

答：使用 v-on 指令来实现事件绑定。

## vue3编译流程？

vue3的编译分两部分：编译期和运行期。编译期间，先通过vue-loader把单文件组件编译成符合原生JavaScript语法的模块；运行期间，会通过内置的模板引擎对模板进行编译，生成render()函数，且按照先父组件后子组件的顺序对模板进行编译。

## vue3内的宏是什么？和语法糖的区别？

按照vue3官方的说法，defineProps() 、defineEmits()、defineExpose()都是宏，它们使用形式上和函数调用差不多，区别是，第一，这些宏不需要通过模块导入；第二，这些宏会在编译期被编译成符合JavaScript原生语法的代码。语法糖也会被编译成符合JavaScript原生语法的代码，vue3官方对这两者进行区分是因为，语法糖创建的是不曾有过的语法，而宏指的是给现有语法新的含义。

## <script setup/>的实现原理？

<script setup/>本质是setup()函数的语法糖，vue-loader在编译期间会把<script setup/>内的代码编译成setup()函数，把defineExpose()内指定的变量编译为setup()函数的返回值。

## 组件实例是什么？

每个单文件组件都有一个组件实例，组件实例是用来存放组件状态的，可以在setup()函数内，通过getCurrentInstance()获取组件实例。

## 启动程序的时候，vue3单文件组件的初始化流程？

单文件组件由组件实例、相应vnode、相应node三部分组成。初始化的时候，先创建组件的实例；接着用父组件vnode初始化实例的props、attrs、slots成员；然后执行setup()函数；接着把模板编译成render()函数；接着创建vnode；如果当前组件有子组件，则切换到子组件，重复上面的步骤；子组件都初始化完成后，把vnode同步到node上。

## createApp()，mount()做了什么工作？

程序在初始化的时候会依次调用createApp()和mount()函数，createApp()负责执行平台相关的代码，mount()负责执行平台无关的代码。createApp()主要做了两件事，第一，创建一个渲染器，它可用于把vdom转化为平台相关的界面代码；第二，返回一个App实例。mount()的作用就是，创建组件实例、vdom、完成依赖收集、并通过渲染器把vdom渲染成平台相关的界面代码。

## 组件的生命周期？

单个组件的生命周期：初始化阶段，先执行setup函数初始化实例，接着执行onBeforeMount，最后执行onMounted；运行阶段，先执行onBeforeUpdate，再执行onUpdated；关闭阶段，先执行onBeforeUnmount，再执行onMounted。

父子组件的生命周期：初始化阶段，先执行父组件的onBeforeMount，然后执行子组件的onBeforeMount，接着执行子组件的onMounted，最后执行父组件的onMounted；运行阶段、关闭阶段同理可推。

## 什么是虚拟DOM？diff算法？

虚拟dom是用JavaScript语言描述的一个树形结构，用于缓存发生变化的响应式数据，当到达下次刷新周期的时候，一次性把所有发生变化的数据同步到实际dom上。diff算法是同步vdom和dom数据的算法，采用就地复用的策略实现高效同步。

## 任务调度系统？

vue3内部维持了3类队列：pre队列——组件更新前置任务队列；queue队列——组件更新时的任务队列；post队列——组件更新后置任务队列。vue3内所有异步任务都会被推进这些队列，当本次宏任务执行结束的时候，会执行一个名为flushJob的微任务，该微任务会按照pre>queue>post的优先级，依次执行每个队列中的函数。

## 异步刷新是什么？怎么实现的？

异步刷新指的是在响应式数据发生更改后，只更新vdom，不立即更新dom，当下一个刷新周期到来的时候，次性把所有发生变化的数据同步到实际dom上。

异步刷新是通过job队列+Promise+diff算法实现的，当响应式数据发生更改时候，会立即更新vdom上的数据，待下个刷新周期到来的前，开始执行flushJob微任务，flushJob会通过diff算法完成vdom和dom之间的数据同步。

## nextTick()的实现原理？

nextTick()作用是，让回调函数在下一次刷新DOM的时候，具体来说在onUpdated后执行。vue3内的nextTick()是通过Promise实现，当微任务flushJob把Job队列中的所有函数都执行完了后，nextTick()回调会被推入微任务队列，然后开始执行。

## 有哪三种Effect？作用？

vue3组件内有3类Effect：computedEffect、watchEffect、renderEffect。正是通过这些Effect才实现数据可监听函数的绑定，其中computedEffect、watchEffect在setup()期间创建，而renderEffect在onBeforeMount和onMounted之间创建。

## watch()，watchEffect()，computed()的区别？

三者都用于实现数据和监听函数的绑定。原理上来说，三者的共同点是，都会在setup期间、即onBeforeMount之前，创建一个Effect对象，并执行依赖收集函数进行依赖收集。区别是，第一，watchEffect、computed的依赖收集函数和监听函数是同一个函数，而watch的依赖收集函数和回调函数是两个函数，这就导致了watchEffect和computed参数中的监听函数会在程序初始化期间执行一次，而watch参数中的监听函数默认不会执行；第二，computed参数中的监听函数在变量发生变化后立即执行，watchEffect、watch参数中的监听函数执行时机可由用户指定，可以立即执行，也可以异步执行。

## 响应式编程的实现原理？

响应式编程的核心是通过发布订阅模式、Effect对象、重写get函数实现的。发布订阅模式实现数据和监听函数的绑定，Effect对象内封装有监听函数和调度策略，get函数则负责进行依赖收集和返回数据。

## ref()和reactive()的区别？

两者都用于实现响应式变量，区别在于reactive()只可以封装引用类型，ref()可以封装任意类型；reactive()通过Proxy实现，ref()通过类封装，并重写get/set实现。就目前而言，要访问ref()封装的变量，需要通过value函数获取。

(16)ref()和ref(null)的区别？

ref()的作用是封装普通数据，实现响应式数据；ref(null)的作用是获取子组件实例。

## attrs和props的区别？

attrs和props的作用都是，以模板属性的形式，从父组件传递数据给子组件。区别是，子组件中获取从父组件传过来的props数据，首先需要用defineProps()声明，而获取attrs数据不需要声明。props的优势是增加了代码可读性，attrs的优势是传值更灵活、代码更简洁。

## provide()和inject()实现原理？

在祖先组件里通过provide()提供的数据，可以在任意子孙组件里通过inject()获取。它是通过原型链实现的。

## emit()的实现原理？

emit()是通过发布订阅模式实现的，子组件发送一个事件，父组件收到该事件会立即执行相应的函数。

## 单向数据流的原因？

单向数据流指的是在 vue中约定，父组件传递给子组件的数据只能单向流动。也就是说，当一个父组件将其数据传递给一个子组件时，子组件不能直接修改该数据，而是需要通过向父组件发送事件来请求更改该数据。这种单向数据流的方式可以确保应用程序的数据流向清晰明确，易于理解和维护。

## 模板的编译流程？在什么时候编译？

模板会先被解析成AST对象，然后通过遍历AST对象生成渲染函数。这个过程发生在setup()函数执行结束后、onBeforeMount之前。

## 组件注册是什么？

若要在vue内使用自定义组件，需要先注册组件。组件注册可以看作是一种声明，用于告诉编译器，某个标签名是组件。组件注册分为局部注册和全局注册，全局注册的组件可以在整个程序内使用，局部注册的组件只可以在特定组件内使用。

## 指令的实现原理？

vue3内的指令是，本质上是一系列钩子函数的集合，可以为这些钩子函数指定执行时机，包括：onBeforeMount、onMounted、onBeforeUpdate、onUpdated、onBeforeUnmount和onUnmounted。

## 各种指令的区别和作用？

v-show和v-if的区别：v-show=false的组件会被创建，但是不会显示；v-if=false的组件不会被创建。

v-on和v-bind的区别：v-on用于绑定事件和事件处理函数；v-bind用于绑定变量。

v-text，v-html和插值表达式的区别：插值表达式存在闪屏的问题，而v-text 、v-html不存在该问题；插值表达式和v-text都会直接以字符串的形式显示html，v-html则可以渲染html标签对。

## v-for为什么要加key？v-for为什么不能和v-if一起使用？

v-for要加key是因为，第一，vue的同步算法采用就地复用的策略，在没有key值的情况下，diff算法认为标签类型相同的元素可以进行复用；有key值的情况下，diff算法认为标签类型相同且key值相等的元素才可以复用。第二，vue只在v-model实现了数据双向绑定，其它地方实现的都是单向绑定，<input/>框内的文字发生变化，不会实时同步到数据上。

在vue3中，v-if的优先级高于v-for，导致v-if访问不了v-for中的变量，因此不能把 v-if 和 v-for 同时用在同一个元素上。

## v-model的实现原理？

v-model实现了数据的双向绑定，它是通过响应式数据+监听输入框事件实现的。

## hook是什么？

hook是一种编程封装思想，通过闭包实现，早在ES6诞生之前，这种编程封装思想在JavaScript中应用广泛，只不过在React17出现后，React官方把这种思想命名为hook，vue3借鉴react命名，也把这种封装思想称作hook。

## vue3插件是什么？

vue3内的所谓的插件不是常规意义上的插件，而是指mount()前执行的初始化函数，vue插件通常用来为 vue 添加全局功能。vue3内通过use()函数引入插件，use()的作用就是执行参数对象内的install()函数。

## 什么是异步组件？

网页的渲染流程一般分两部，先从网络加载网页，然后渲染。异步组件指的是，首次加载和渲染网页的时候，不从网络上加载该组件对应的JavaScript文件，而是等到需要显示该组件的时候，再从网络上加载该组件并渲染显示出来。

## 前端路由是什么？什么是路由懒加载？

路由能够根据不同的URL地址返回给用户不同的内容或页面，如果路由位于浏览器端，就是前端路由；如果路由位于服务器端，就是后端路由。

对于SPA单页应用，打包构建时，JavaScript包会变得非常大，会使得页面首次加载的时间变长 。若将不同路由对应的组件分割成不同的文件，然后当路由被访问的时候才从服务器加载对应文件，这就是路由的懒加载。

## v-if 和 v-for 的优先级哪个高？

在 vue2 中 v-for 的优先级更高，但是在 vue3 中优先级改变了。v-if 的优先级更高。