Vue

v-show和v-if区别

- v-show
  - v-show不管条件是真还是假，第一次渲染的时候都会编译出来，也就是标签都会添加到DOM中，之后切换的时候，通过display: none样式来显示隐藏元素。可以说只是改变css的样式，几乎不会影响什么性能。
  - 如果是一些固定的，条件内容都不怎么会改变的，频繁切换的，使用v-show会比较省性能
  - 如果是子组件，每次切换子组件不执行生命周期，使用v-show
- v-if
  - 在首次渲染的时候，如果条件为假，什么也不操作，页面当作没有这些元素。当条件为真的时候，开始局部编译，动态的向DOM元素里面添加元素。当条件从真变为假的时候，开始局部编译，卸载这些元素，也就是删除。
    - v-if 能够控制是否生成虚拟节点，也就间接控制了是否生成对象的dom。当v-if为true时，会生成对应的虚拟节点，并且成对应的dom元素；当其为false时，不会生成对应的vnode，自然不会生成任何dom元素。
  - 如果子组件需要重新执行生命周期，那么使用v-if才能触发。
  - v-if 有配套的 v-else-if 和 v-else
  - 设置v-if=false的时候，会在当前的模板中创建一个注释节点（占位节点）

vuex

- 作用
  - 管理多个组件的共享状态
    - 全局状态（平级）
    - 嵌套组件使用统一数据（子孙级）
    - 兄弟组件维护同一状态
  - 状态变更跟踪
- state
  - 存储状态的地方
- getter
  - 管理派生状态。类似于computed,依赖变了之后才会导致派生数据的变化
  - 依赖数据是state和其他getter中的属性
    - 在组件中通过this.$store.getters.xxx的方式来访问getter
- 核心属性
  - mutation
    - 改变Vuex中的状态的唯一途径就是显式地提交 (commit) mutation
    - 同步
  - action
    - 异步
    - Action 提交的是 mutation，而不是直接变更状态
      - action的出现是为了弥补mutation不能异步的操作的缺陷
      - 即同步操作直接提交mutation，异步操作使用action提交mutation
  - module
    - 将store分割成module,来做模块化
    - 通过rootState、rootGetters来访问全局状态和派生状态
- 其他属性
  - mapState
    - 可以在组件中批量使用state中的状态

Vue3

- 兼容vue2
- 与Vue2响应式原理区别
  - Vue2
    - Object.defignProperty
      - 1. 深度监听在初始化时就一次性递归监听好了
      - 2. 无法跟踪到对象增加新值、删除属性
      - 3. 数组需要额外处理
  - Vue3
    - proxy
      - 1. 深度监听是在获取值的时候才对其监听(按需监听)
      - 2. 可以很完备的监听到属性增加和删除。
      - 3. 数组可以原生使用api,不需要再对需要监听的api再做处理
- 有哪些性能优化
  - diff算法优化
    - 在Vue3的做法是将静态的元素（节点），会做静态提升，只会被创建一次，在渲染时直接复用
    - 这样的话再次更新时候就不用比较静态节点了
    - 关于这个静态标记，其作用是为了会发生变化的地方添加一个flag标记，下次发生变化的时候直接找该地方进行比较

compiler-core
transformHoist.ts
export const isStaticExpressionMap = new Map<
export function isStaticExp(p: JSNode): boolean {
export function getConstantType(node: SimpleExpressionNode, ast: RootNode): ConstantTypes {
export function hoistStatic(root: RootNode, context: TransformContext) {
function walk(children: BlockIRNode['dynamic'], parent: BlockIRNode, info: DynamicChild
function walkChildren(parent: ParentNode, parentCallback: NodeTransformCallback
function hoistNode(node: JSChildNode | FunctionExpression | VNodeCall) {
function getPatchFlag(node: JSChildNode): number | undefined {
function getGeneratedPropsConstantType(node: PlainElementNode, context: TransformCo
function getNodeProp(node: PlainElementNode, name: string, dynamicOnly: boolean =
function getVNodeBlockHelper(isSlotted: boolean, isDynamic: boolean) {
function getVNodeHelper(ssr: boolean, isComponent: boolean) {
function addHoisted(exp: JSChildNode): SimpleExpressionNode {
function isSingleElementRoot(root: RootNode, child: CommonChildNode) {
function stringifyExpression(exp: ExpressionNode | string): string {
function isMemberExpressionBrowser(path: string): boolean {
function isMemberExpressionNode(path: string, context: CompilerCoreOptions): boolean
function isMemberExpression(path: string, context: CompilerCoreOptions): boolean {
function assertExp(exp: ExpressionNode | undefined) {
function assertAccess(path: string, type: AccessTypes) {
function assertObjectPatternElement(id: Node, e: Node, inLoop: boolean) {
function assertFunctionPramsNoPrefix(1: Node, inLoop: boolean) {
function assertFunctionBodyNoPrefix(1: Node, inLoop: boolean) {
function assertNodeNoPrefix(1: Node, inLoop: boolean) {
function assertNodeListNoPrefix(1: Node[], inLoop: boolean) {
function assertNodeMaybeNullNoPrefix(1: Node | null, inLoop: boolean) {
function assertNodeMaybeNullListNoPrefix(1: Node[] | null, inLoop: boolean) {
function assertNodeMaybeNullOrPatternNoPrefix(1: Node | null, inLoop: boolean) {
function assertNodeMaybeNullOrPatternListNoPrefix(1: Node[] | null, inLoop: boolean)
function assertNodeOrPatternNoPrefix(1: Node, inLoop: boolean) {
function assertNodeOrPatternListNoPrefix(1: Node[], inLoop: boolean) {
function assertNodeOrPattern(1: Node, inLoop: boolean) {
function assertNodeListOrPattern(1: Node[], inLoop: boolean) {
function assertNodeMaybeNullOrPattern(1: Node | null, inLoop: boolean) {
function assertNodeMaybeNullListOrPattern(1: Node[] | null, inLoop: boolean)
function assertNodeOrPatternOrSpecialNode(1: Node, inLoop: boolean) {
function assertNodeListOrPatternOrSpecialNode(1: Node[], inLoop: boolean) {
function assertNodeMaybeNullOrPatternOrSpecialNode(1: Node | null, inLoop: boolean)
function assertNodeMaybeNullListOrPatternOrSpecialNode(1: Node[] | null, inLoop: b
function assertNodeOrPatternOrSpecialNodeOrFunctionLike(1: Node, inLoop: boolean)
function assertNodeListOrPatternOrSpecialNodeOrFunctionLike(1: Node[], inLoop: boolean
function assertNodeMaybeNullOrPatternOrSpecialNodeOrFunctionLike(1: Node | null, in
function assertNodeMaybeNullListOrPatternOrSpecialNodeOrFunctionLike(1: Node[] | nu
function assertNodeOrPatternOrSpecialNodeOrFunctionLikeOrClass(1: Node, inLoop: boo
function assertNodeListOrPatternOrSpecialNodeOrFunctionLikeOrClass(1: Node[], inLoo
function assertNodeMaybeNullOrPatternOrSpecialNodeOrFunctionLikeOrClass(1: Node | n
function assertNodeMaybeNullListOrPatternOrSpecialNodeOrFunctionLikeOrClass(1: Node
function assertNodeOrPatternOrSpecialNodeOrFunctionLikeOrClassOrExpression(1: Node,
function assertNodeListOrPatternOrSpecialNodeOrFunctionLikeOrClassOrExpression(1: N
function assertNodeMaybeNullOrPatternOrSpecialNodeOrFunctionLikeOrClassOrExpression
function assertNodeMaybeNullListOrPatternOrSpecialNodeOrFunctionLikeOrClassOrExpres
function assertNodeOrPatternOrSpecialNodeOrFunctionLikeOrClassOrExpressionOrStatem
function assertNodeListOrPatternOrSpecialNodeOrFunctionLikeOrClassOrExpressionOrSt
function assertNodeMaybeNullOrPatternOrSpecialNodeOrFunctionLikeOrClassOrExpressio
function assertNodeMaybeNullListOrPatternOrSpecialNodeOrFunctionLikeOrClassOrExpre