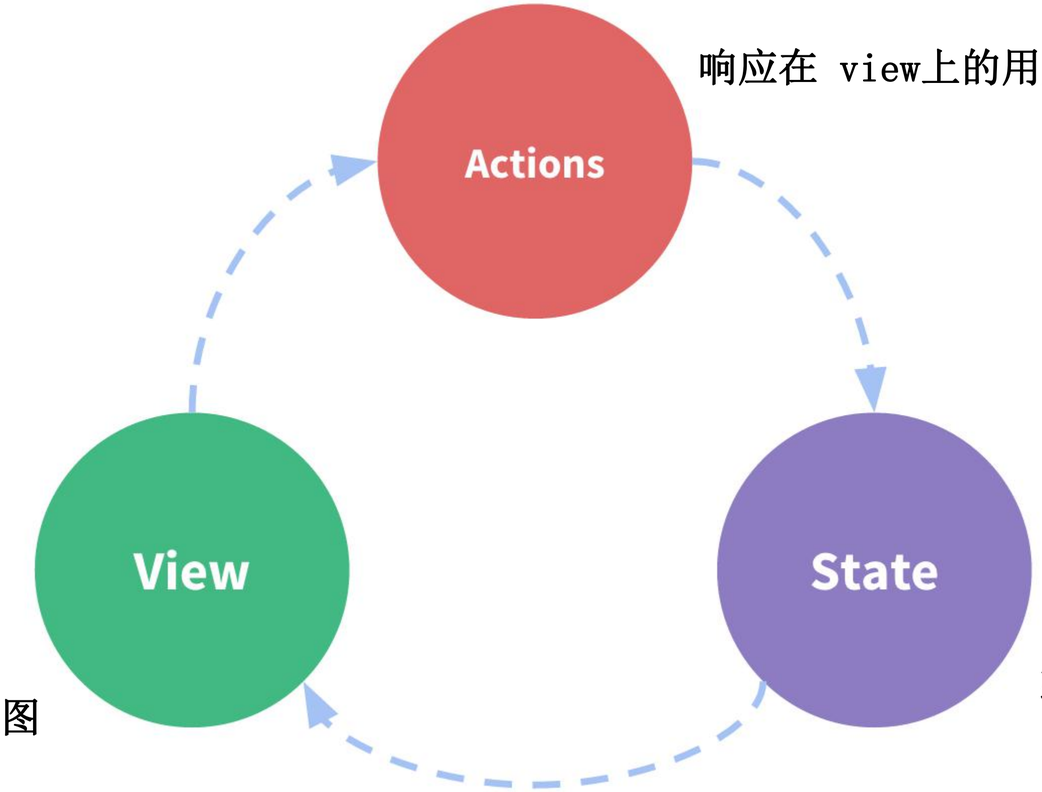
生态篇

Vuex

为什么需要 Vuex

Vuex 是一种状态管理模式

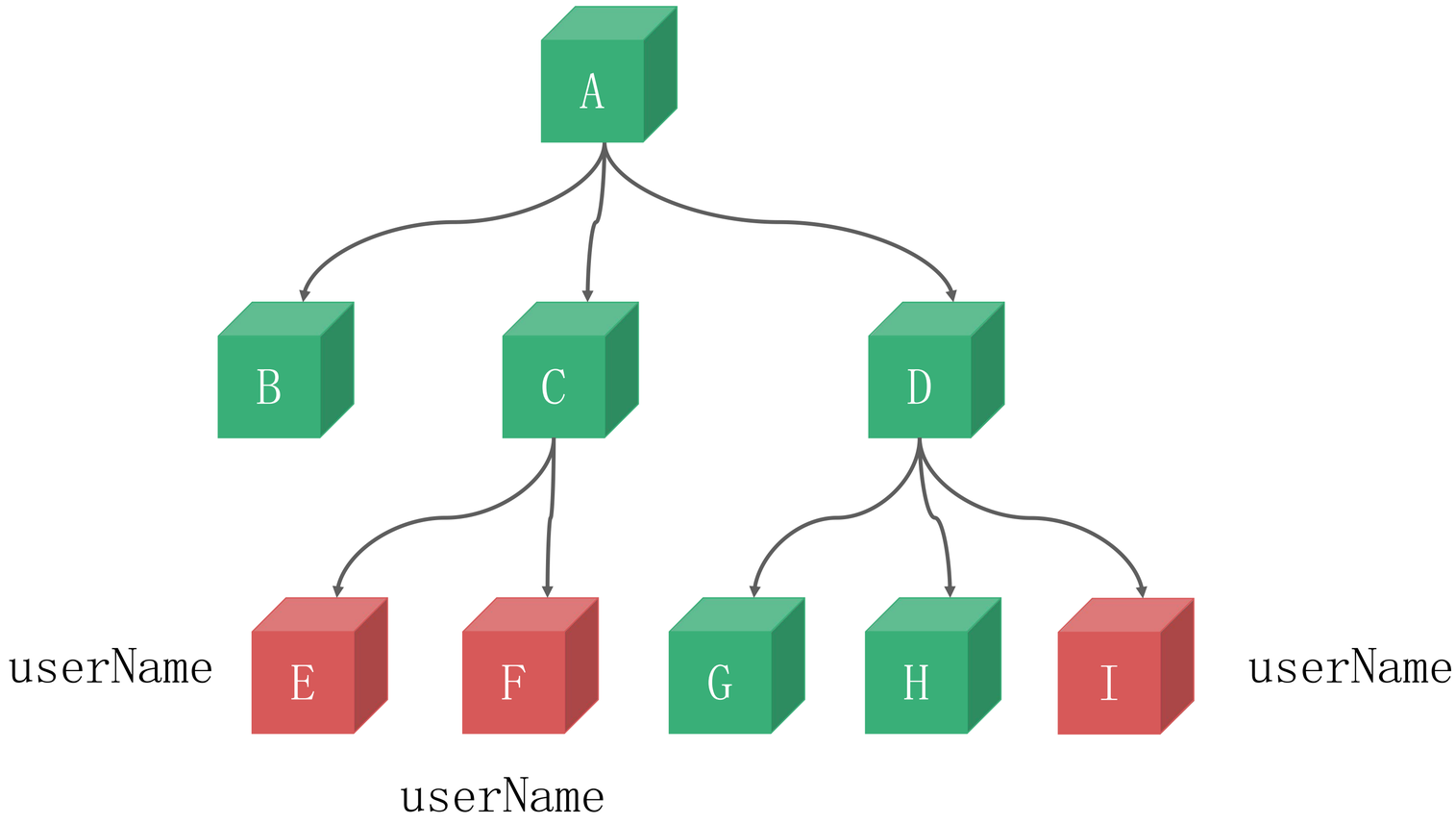


响应在 view上的用户输入导致的状态变化

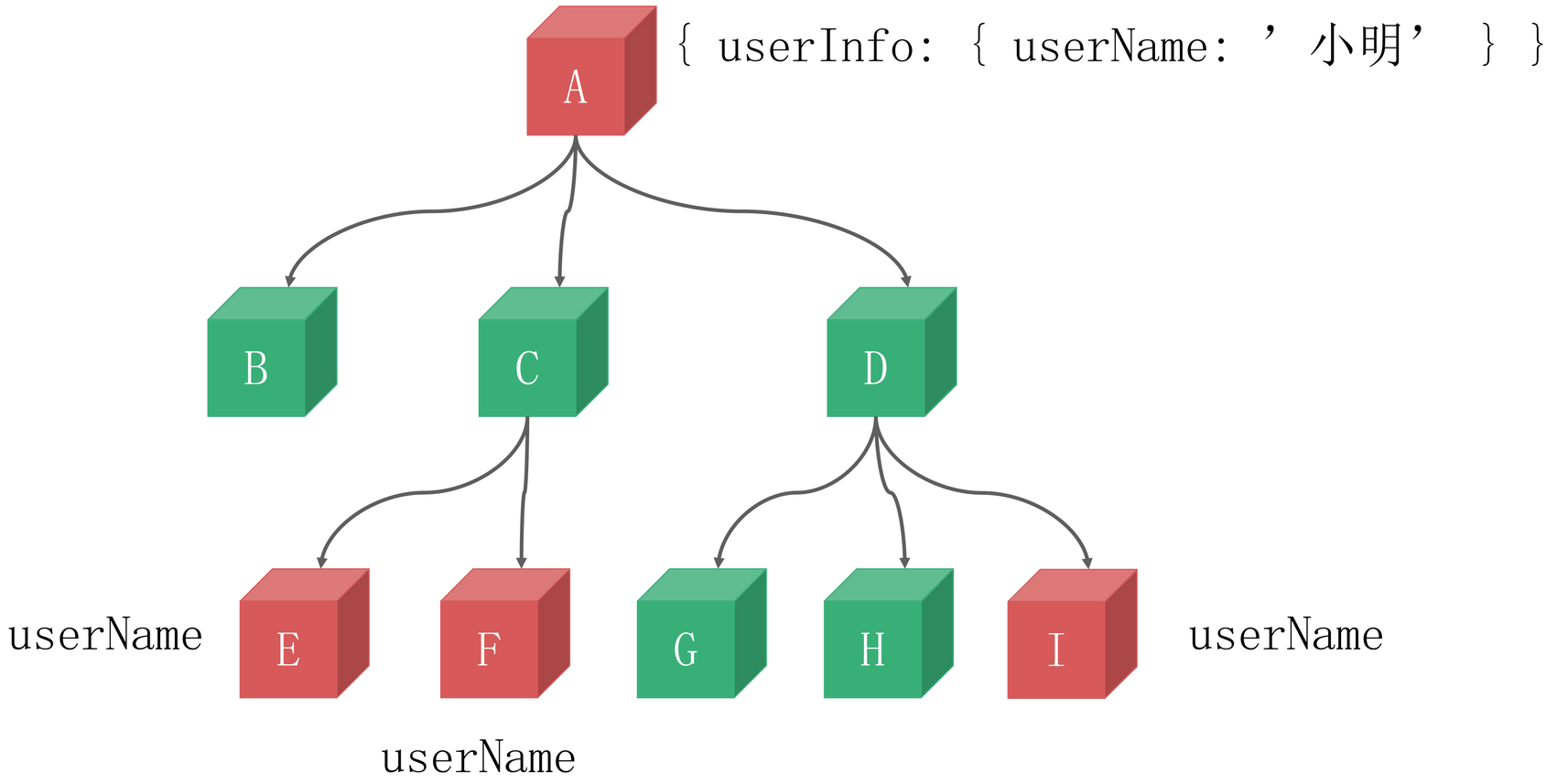
以声明方式将 state 映射到视图

驱动应用的数据源

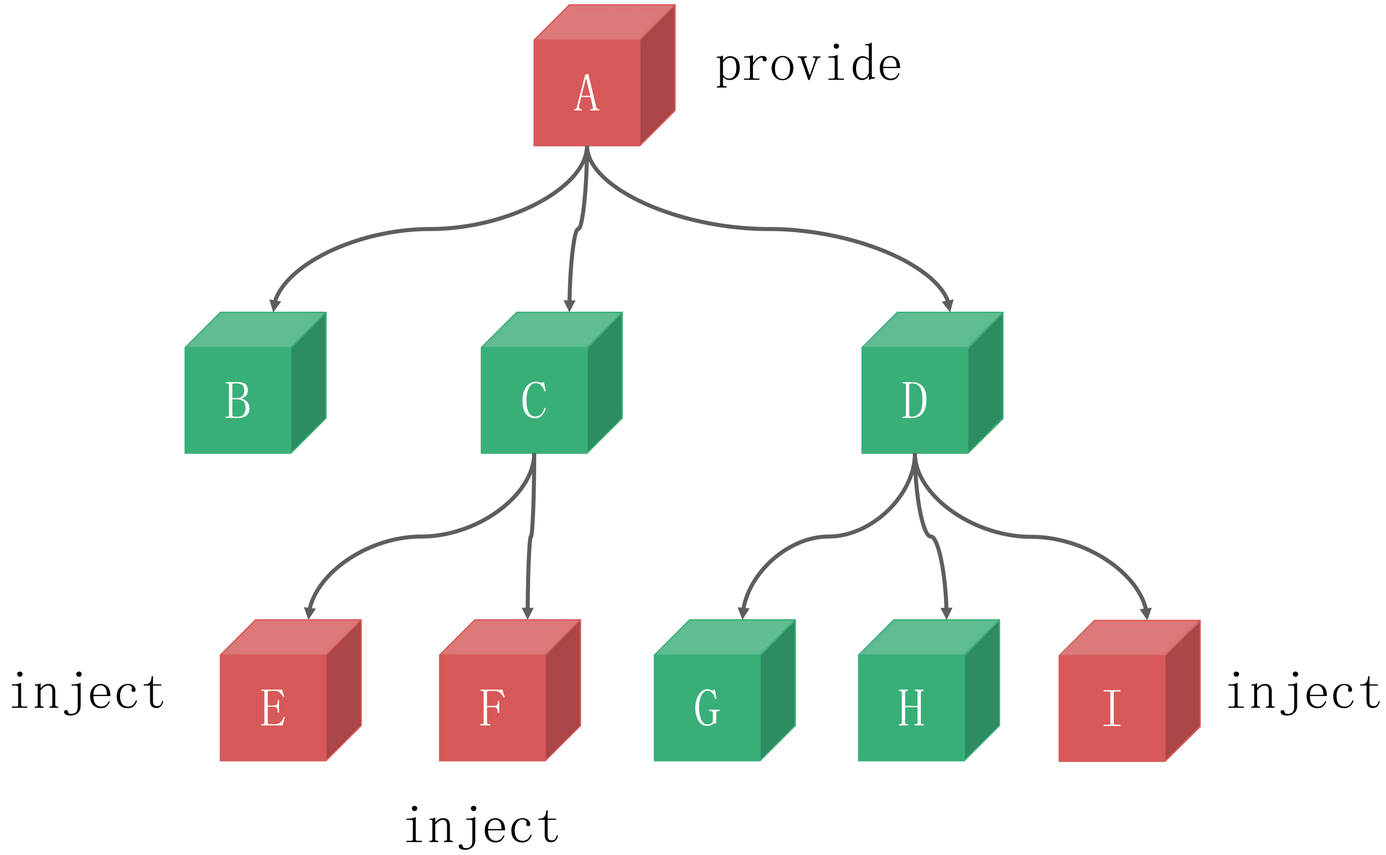
状态管理



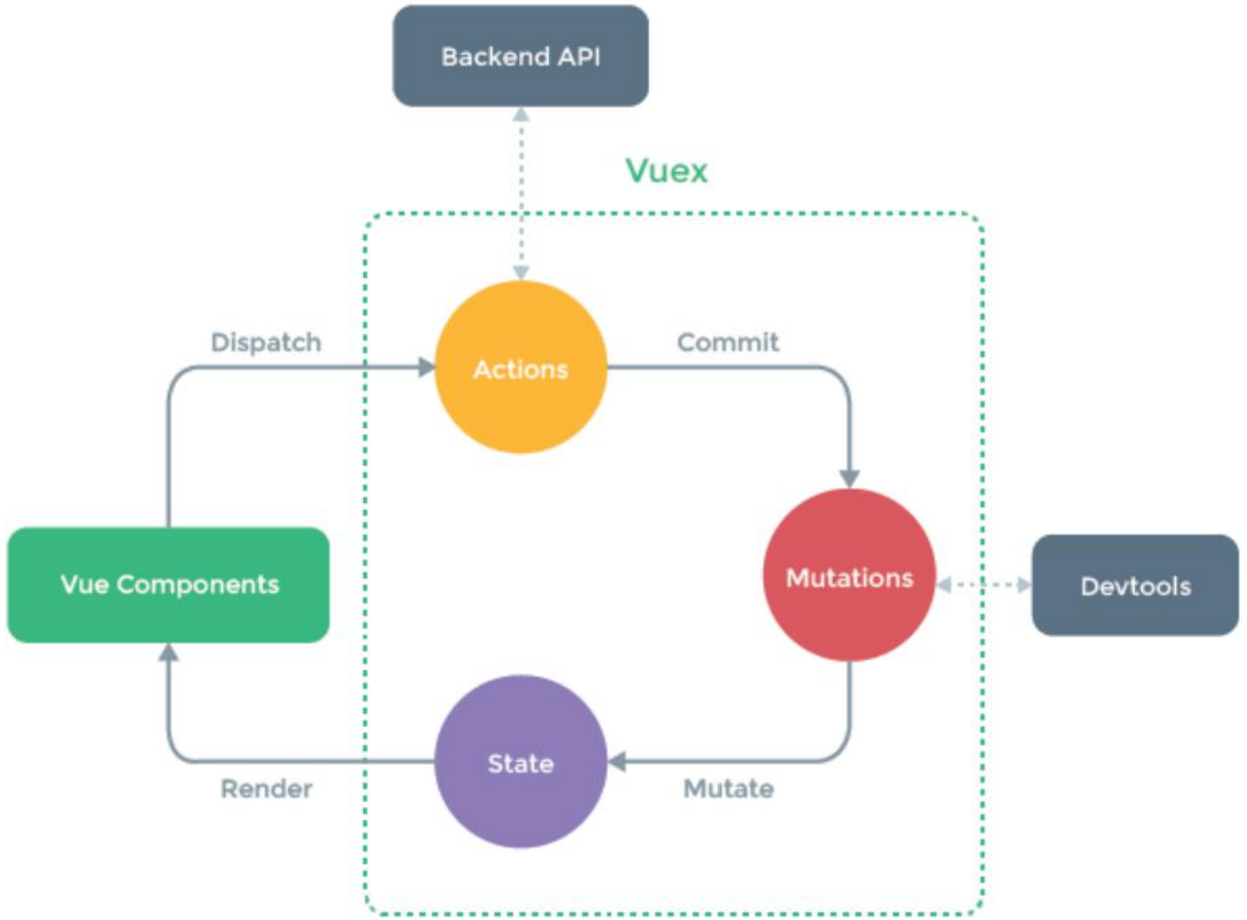
状态管理


A
{ userInfo: { userName: ’小明’ } }
B
C
D
userName
E
F
G
H
I
userName
userName

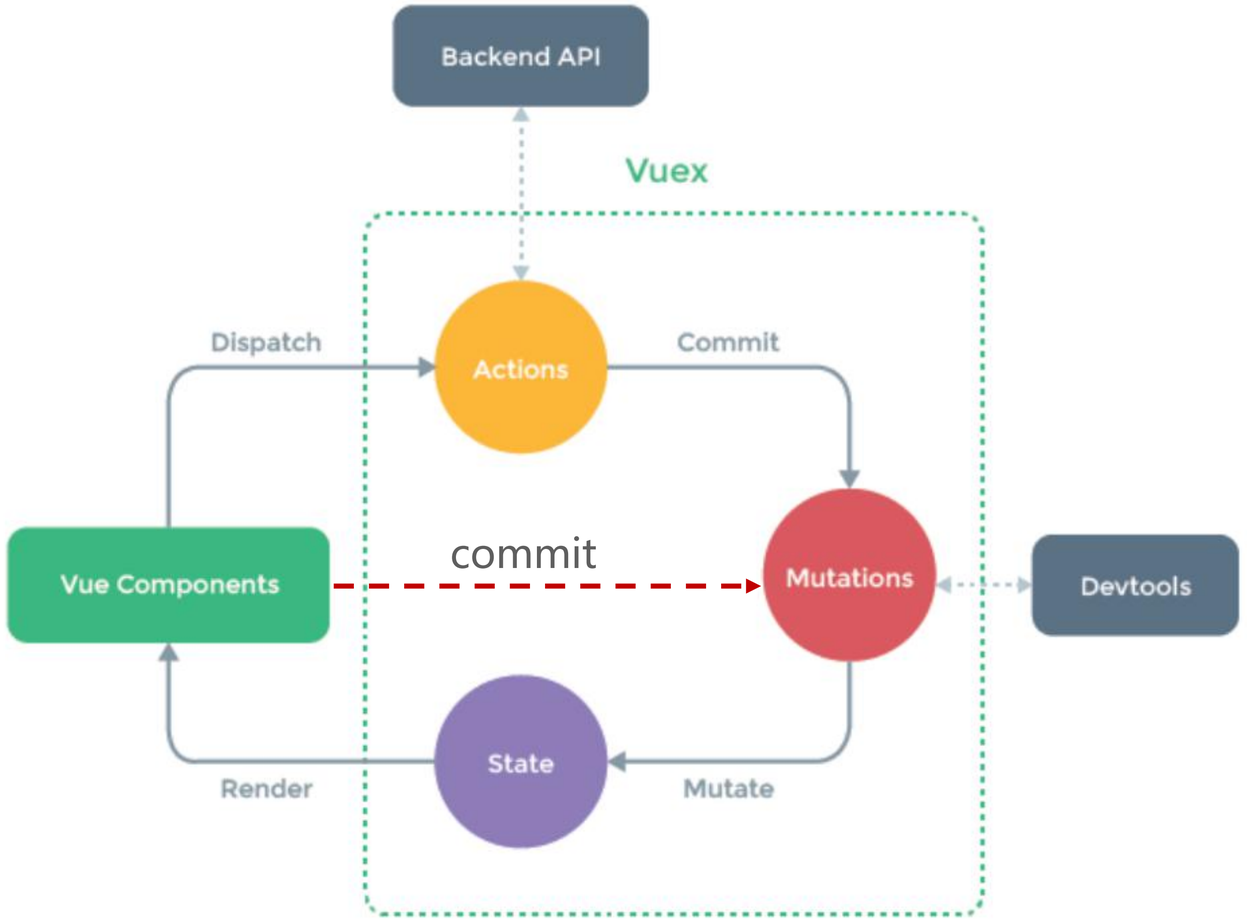
provide/inject



Vuex 运行机制


Backend API
Vuex
Dispatch
Actions
Commit
Vue Components
Mutations
Devtools
State
Render
Mutate

Vuex 运行机制


Backend API
Vuex
Dispatch
Actions
Commit
commit
Vue Components
Mutations
Devtools
State
Render
Mutate

课后习题

- Vuex 是通过什么方式提供响应式数据的？

如何在 Vue 中使用 Vuex

创建 store

- 初始 state 状态对象
- 提供 mutation 方法

```
Vue.use(Vuex)
const store = new Vuex.Store({
  state: {
    count: 0
  },
  mutations: {
    increment (state) {
      state.count++
    }
  }
})
```

课后习题

- $store 是如何是如何挂载到实例 this 上的

Vuex 的核心概念和底层原理

核心概念

- State —— this.$store.state.xxx　取值
- Getter —— this.$store.getters.xxx 取值
- Mutation —— this.$store.commit(“xxx”) 赋值
- Action —— this.$store.dispatch(“xxx”) 赋值
- Module

底层原理

- State：提供一个响应式数据
- Getter: 借助 Vue 的计算属性 computed 来实现缓存
- Mutation: 更改 state 方法
- Action: 触发 mutation 方法
- Module: Vue.set 动态添加state到响应式数据中

课后习题

- 扩展简化版的 min-vuex，实现 getters，并实现 Vuex 的方式注入$store

Vuex 最佳实践

核心概念

- State —— this.$store.state.xxx —— mapState 取值
- Getter —— this.$store.getters.xxx —— mapGetters 取值
- Mutation —— this.$store.commit(“xxx”) —— mapMutations 赋值
- Action —— this.$store.dispatch(“xxx”) —— mapActions 赋值
- Module

使用常量替代 Mutation 事件类型

```js
// mutation-types.js
export const SOME_MUTATION = 'SOME_MUTATION'
```

```js
// store.js
import Vuex from 'vuex'
import { SOME_MUTATION } from './mutation-types'

const store = new Vuex.Store({
  state: { ... },
  mutations: {
    // 我们可以使用 ES2015 风格的计算属性命名功能来使用一个常量作为函数名
    [SOME_MUTATION] (state) {
      // mutate state
    }
  }
})
```

Module

- 开启命名空间 namespaced: true
- 嵌套模块不要过深，尽量扁平化
- 灵活应用 createNamespacedHelpers

课后习题

- 扩展购物车示例，提供单次添加 1-N 的数量到购物车的功能

Vue Router 使用场景和使用方式

传统开发模式

- www.xxx.com —— index.html
- www.xxx.com/about —— about.html
- www.xxx.com/xxx —— xxx.html

单页面(SPA)开发模式

- www.xxx.com —— index.html
- www.xxx.com/about —— index.html
- www.xxx.com/xxx —— index.html

解决的问题

- 监听 URL 的变化，并在变化前后执行相应的逻辑
- 不同的 URL 对应不同的不同的组件
- 提供多种方式改变 URL 的 API（URL 的改变不能导致浏览器刷新）

使用方式

- 提供一个路由配置表，不同 URL 对应不同组件的配置
- 初始化路由实例 new VueRouter()
- 挂载到 Vue 实例上
- 提供一个路由占位，用来挂载 URL 匹配到的组件

课后习题

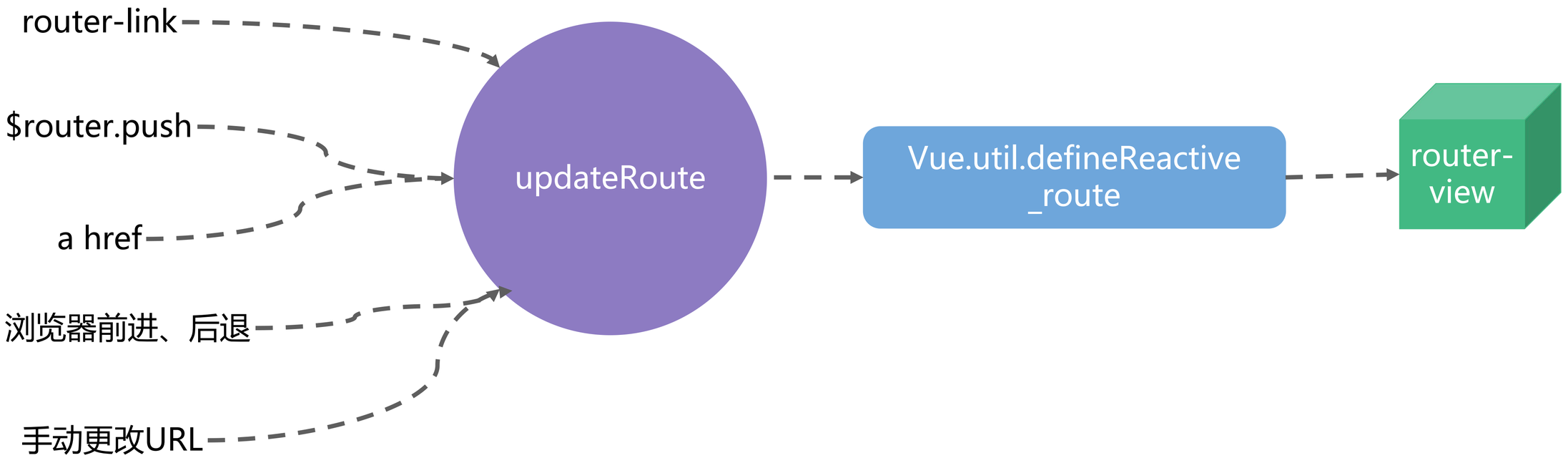
- SPA 的缺点有哪些，如何解决？

路由类型及底层原理

路由类型

- Hash 模式：丑，无法使用锚点定位
- History 模式：需要后端配合，IE9 不兼容（可使用强制刷新处理）

底层原理


router-link
$router.push
a href
浏览器前进、后退
手动更改URL
updateRoute
Vue.util.defineReactive
_route
router-
view

Nuxt 解决了哪些问题

SPA 缺点

- 不利于 SEO
- 首屏渲染时间长

服务端渲染 SSR

预渲染 Prerendering

Prerendering

- 适用于静态站点

about.html

contact.html

xxx.html

SSR

- 动态渲染
- 配置繁琐

Nuxt

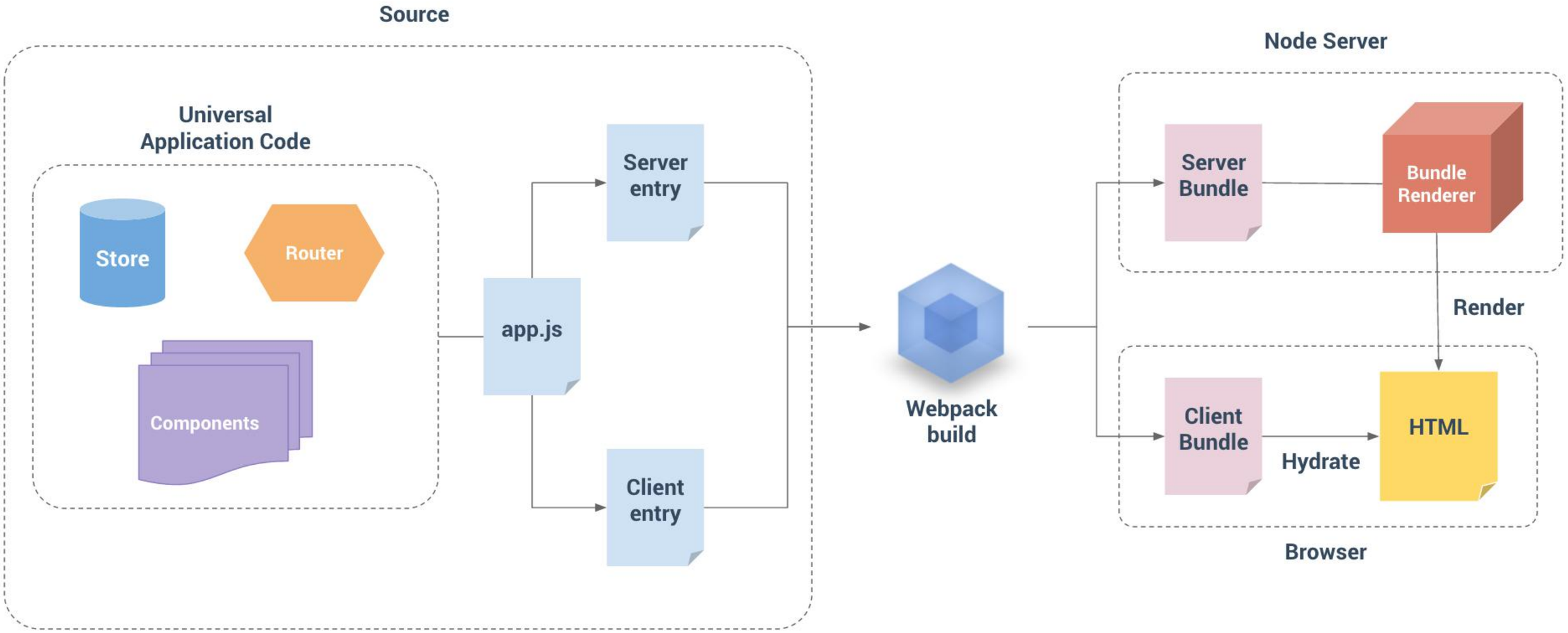
- 静态站点
- 动态渲染
- 简化配置

习题

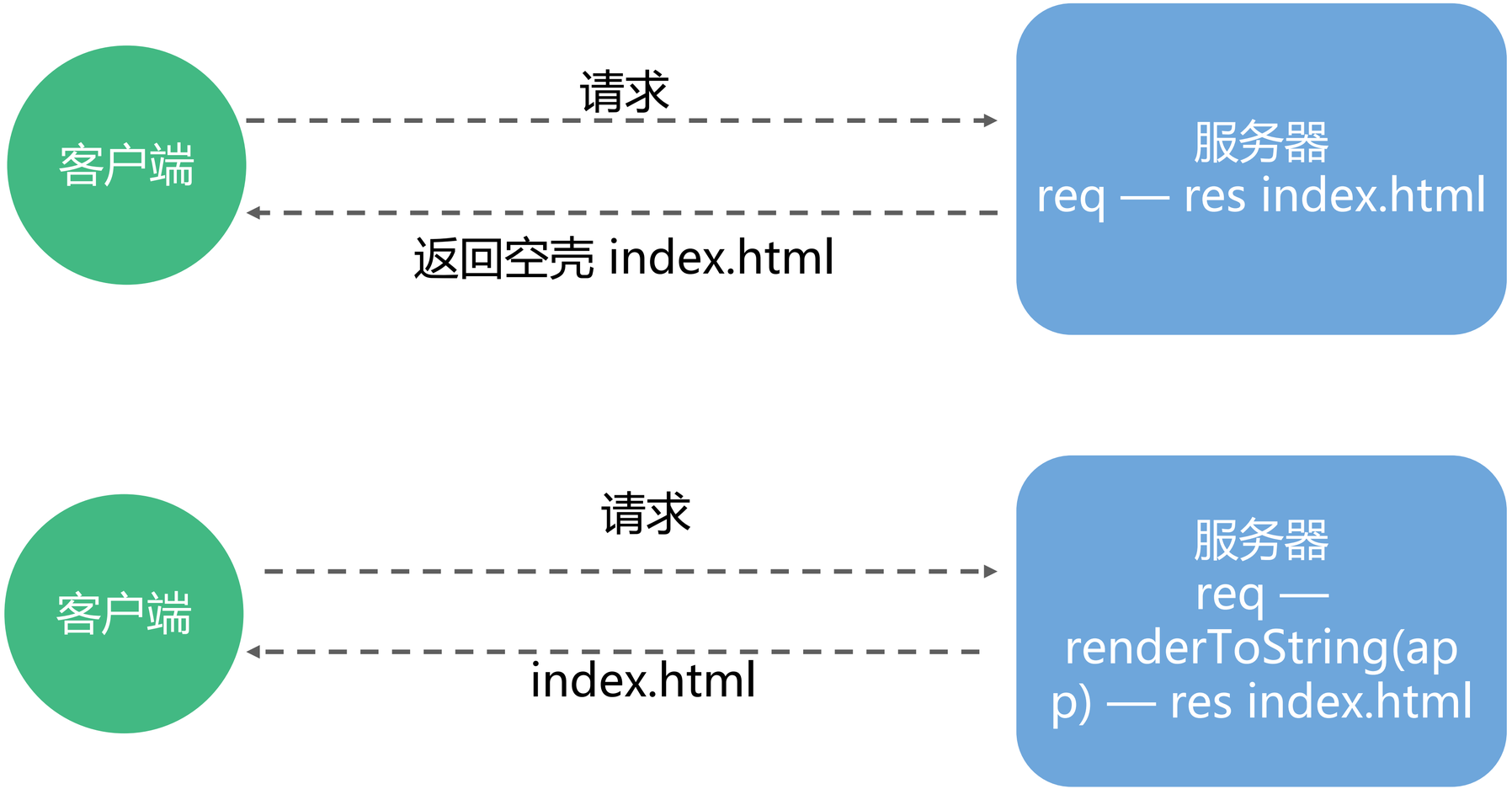
- 对于动态内容，如果不使用 SSR，如何做 SEO

Nuxt 核心原理是什么

SSR


Source
Universal
Application Code
Store
Router
Components
app.js
Server
entry
Client
entry
Webpack
build
Node Server
Server
Bundle
Bundle
Renderer
Render
Client
Bundle
Hydrate
HTML
Browser

流程图


客户端
请求
返回空壳 index.html
服务器
req — res index.html
客户端
请求
index.html
服务器
req — renderToString(app) — res index.html

UI 组件库对比

Element UI vs Ant Design Vue vs iView

	数量	单测	admin	背景	原型、设计
Element UI	46	81%	vue-element-admin（社区）	饿了么	Axure、Sketch
Ant Design Vue	55	87%	Pro（社区）	社区主导、蚂蚁金服技术支持	Axure、Sketch
iView	54	无	iView-admin	TalkingData	无

提升开发效率和体验的常用开发工具：

Vetur、ESLint、Prettier、Vue DevTools

Vetur

- 语法高亮
- Lint 检查
- 格式化

ESLint

- 代码规范
- 错误检查

Prettier

- 格式化

Vue DevTools

- 集成 Vuex
- 可远程调试

单元测试的重要性及其使用

重要性

- 减少 bug
- 提高项目的稳定性
- 提高开发速度

使用方式

- Jest 内置单元测试 Mocha、断言库 chai、JSDOM、测试报告 istanbul
- @vue/test-utils
- vue-test
- babel-jest
- jest-serializer-vue
- sinon

课后习题

- Vuex 是通过什么方式提供响应式数据的？

课后习题

- $store 是如何是如何挂载到实例 this 上的

课后习题

- 扩展简化版的 min-vuex，实现 getters，并实现vuex的方式注入$store

课后习题

- 扩展购物车示例，提供单次添加 1-N 的数量到购物车的功能

课后习题

- SPA 的缺点有哪些，如何解决？

习题

- 对于动态内容，如果不使用 SSR，如何做 SEO