Vue训练营

大圣@开课吧

关于我

- 关注老司机不迷茫
- 掘金:https://juejin.im/user/59532176f265da6c317d8e14
- 知乎:https://www.zhihu.com/people/woniuppp
- github: https://github.com/shengxinjing/vue-master(训练营代码会随时同步到github)

Vue源码

- 为什么要看源码
- 如何阅读源码
- 小技巧
- 实战阅读
- Vue3

认知的境界

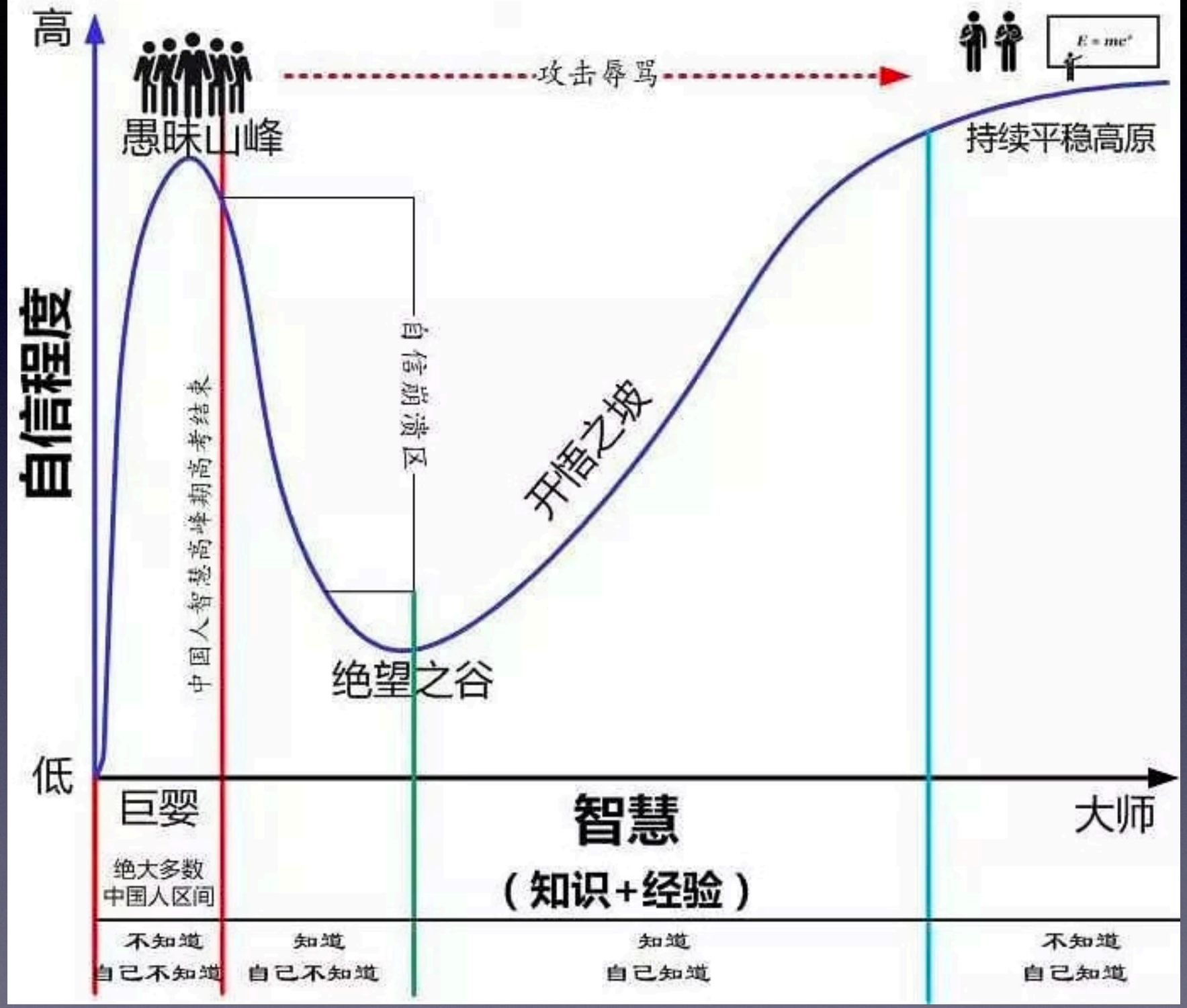
知道自己不知道

不知道自己不知道

知道自己知道

不知道自己知道

邓宁-克鲁格心理效应（Dunning–Kruger effect）



Vue3发布beta版

- 距离正式发布还有些时间，短期内依然是vue2
- vue2和vue3原理相通的，关注更新的部分即可
  - 直接升级
  - Proxy取代defineProperty
  - vdom重构
  - 对外测试版本，迭代后还有rc版 然后才是release
  - 拥抱Composition

Why

- 面试驱动
- 成长驱动
- 视野

How

- 通读官网
- package.json入手，找到起点
- 主线优先
- 测试代码辅助
- console和断点调试

搞起

- 目录
- 渲染流程
- 组件化
- 虚拟dom
- 编译
- 工具
- router和vuex

基础

- defineProperty
- Proxy
- vdom
- 模板编译

- 1. 项目目录
  - benchmarks 性能
  - dist: 构建后文件的输出目录
  - flow: 类型声明，使用开源项目 [Flow]
  - packages: 存放独立发布的包的目录
  - scripts: 构建相关的文件
    - git-hooks：存放git钩子的目录
    - alias.js：别名配置
    - config.js：生成rollup配置的文件
    - build.js：对 config.js 中所有的rollup配置进行构建
    - ci.sh：持续集成运行的脚本
    - release.sh： 用于自动发布新版本的脚本
  - examples: 存放一些使用Vue开发的应用案例
  - test: 包含所有测试文件
  - src: 源码,重点
    - compiler: 编译器代码的存放目录，将 template 编译为 render 函数
    - core: 核心代码 ， 与平台无关的代码
      - observer： 响应系统，包含数据观测的核心代码
      - vdom：包含虚拟DOM创建(creation)和打补丁(patching)的代码
      - instance：包含Vue构造函数设计相关的代码
      - global-api：包含给Vue构造函数挂载全局方法(静态方法)或属性的代码
      - components：包含抽象出来的通用组件
    - platforms: 不同平台的支持
      - web：web平台
      - weex：混合应用
    - serve: 服务端渲染，包含(server-side rendering)的相关代码
    - sfc: 包含单文件组件(.vue 文件)的解析逻辑，用于vue-template-compiler包
    - shared: 共享代码，包含整个代码库通用的代码

寻找入口

- package.json
- build.js
- config.js
- web-runtime-with-compile.js

一路寻找

- Web-entry-with-compiler 扩展$mount
- runtime/index 扩展path，没有compile的$mount
- core/index 扩展全局api(use, extend)
- instance/index 终于，new vue就是执行了这个init

```
1   import { initMixin } from './init'
2   import { stateMixin } from './state'
3   import { renderMixin } from './render'
4   import { eventsMixin } from './events'
5   import { lifecycleMixin } from './lifecycle'
6   import { warn } from '../util/index'
7
8   function Vue (options) {
9     if (process.env.NODE_ENV !== 'production' &&
10      !(this instanceof Vue)
11    ) {
12      warn('Vue is a constructor and should be called with the `new` keywo
13    }
14    this._init(options)
15  }
16
17  initMixin(Vue)
18  stateMixin(Vue)
19  eventsMixin(Vue)
20  lifecycleMixin(Vue)
21  renderMixin(Vue)
22
23  export default Vue
```

initMixin

- 初始化生命周期
- 初始化事件
- 初始化渲染
- 初始化state(data,props,computed)

```
14
15  export function initMixin (Vue: Class<Component>) {
16    Vue.prototype._init = function (options?: Object) {
17      const vm: Component = this
18      // a uid
19      vm._uid = uid++
20
21      let startTag, endTag
22      /* istanbul ignore if */
23      if (process.env.NODE_ENV !== 'production' && config.performance && mark) {
24        startTag = `vue-perf-start:${vm._uid}`
25        endTag = `vue-perf-end:${vm._uid}`
26        mark(startTag)
27      }
28
29      // a flag to avoid this being observed
30      vm._isVue = true
31      // merge options
32      if (options && options._isComponent) {
33        // optimize internal component instantiation
34        // since dynamic options merging is pretty slow, and none of the
35        // internal component options needs special treatment.
36        initInternalComponent(vm, options)
37      } else {
38        vm.$options = mergeOptions(
39          resolveConstructorOptions(vm.constructor),
40          options || {},
41          vm
42        )
43      }
44      /* istanbul ignore else */
45      if (process.env.NODE_ENV !== 'production') {
46        initProxy(vm)
47      } else {
48        vm._renderProxy = vm
49      }
50      // expose real self
51      vm._self = vm
52      initLifecycle(vm)
53      initEvents(vm)
54      initRender(vm)
55      callHook(vm, 'beforeCreate')
56      initInjections(vm) // resolve injections before data/props
57      initState(vm)
58      initProvide(vm) // resolve provide after data/props
59      callHook(vm, 'created')
60
61      /* istanbul ignore if */
62      if (process.env.NODE_ENV !== 'production' && config.performance && mark) {
63        vm._name = formatComponentName(vm, false)
64        mark(endTag)
```

```
    // expose real self
    vm._self = vm
    initLifecycle(vm)
    initEvents(vm)
    initRender(vm)
    callHook(vm, 'beforeCreate')
    initInjections(vm) // resolve injections before data/props
    initState(vm)
    initProvide(vm) // resolve provide after data/props
    callHook(vm, 'created')

    /* istanbul ignore if */
    if (process.env.NODE_ENV !== 'production' && config.performance && mark) {
      vm._name = formatComponentName(vm, false)
      mark(endTag)
      measure(`vue ${vm._name} init`, startTag, endTag)
    }

    if (vm.$options.el) {
      vm.$mount(vm.$options.el)
    }
  }
}
```

lifeCycle

```
export function initLifecycle (vm: Component) {
  const options = vm.$options

  // locate first non-abstract parent
  // 初始化parent.$children
  let parent = options.parent
  if (parent && !options.abstract) {
    while (parent.$options.abstract && parent.$parent) {
      parent = parent.$parent
    }
    parent.$children.push(vm)
  }

  vm.$parent = parent
  vm.$root = parent ? parent.$root : vm

  vm.$children = []
  vm.$refs = {}

  vm._watcher = null
  vm._inactive = null
  vm._directInactive = false
  vm._isMounted = false
  vm._isDestroyed = false
  vm._isBeingDestroyed = false
}
```

initEvents

```
export function eventsMixin (Vue: Class<Component>) {
  const hookRE = /^hook:/
  // $on $emit $oncw 经典的发布订阅模式的实践
  Vue.prototype.$on = function (event: string | Array<string>, fn: Function): Component {…
  }

  Vue.prototype.$once = function (event: string, fn: Function): Component {…
  }

  Vue.prototype.$off = function (event?: string | Array<string>, fn?: Function): Component {…
  }

  Vue.prototype.$emit = function (event: string): Component {…
  }
}
```

initRender

```
export function initRender (vm: Component) {
  vm._vnode = null // the root of the child tree
  vm._staticTrees = null // v-once cached trees
  const options = vm.$options
  const parentVnode = vm.$vnode = options._parentVnode // the placeholder node in parent tree
  const renderContext = parentVnode && parentVnode.context
  vm.$slots = resolveSlots(options._renderChildren, renderContext)
  vm.$scopedSlots = emptyObject
  // bind the createElement fn to this instance
  // so that we get proper render context inside it.
  // args order: tag, data, children, normalizationType, alwaysNormalize
  // internal version is used by render functions compiled from templates
  vm._c = (a, b, c, d) => createElement(vm, a, b, c, d, false)
  // normalization is always applied for the public version, used in
  // user-written render functions.
  vm.$createElement = (a, b, c, d) => createElement(vm, a, b, c, d, true)

  // $attrs & $listeners are exposed for easier HOC creation.
  // they need to be reactive so that HOCs using them are always updated
  const parentData = parentVnode && parentVnode.data

  /* istanbul ignore else */
  if (process.env.NODE_ENV !== 'production') {
    defineReactive(vm, '$attrs', parentData && parentData.attrs || emptyObject, () => {
      !isUpdatingChildComponent && warn(`$attrs is readonly.`, vm)
    }, true)
    defineReactive(vm, '$listeners', options._parentListeners || emptyObject, () => {
      !isUpdatingChildComponent && warn(`$listeners is readonly.`, vm)
    }, true)
  } else {
    defineReactive(vm, '$attrs', parentData && parentData.attrs || emptyObject, null, true)
    defineReactive(vm, '$listeners', options._parentListeners || emptyObject, null, true)
  }
```

initState

```
export function initState (vm: Component) {
  vm._watchers = []
  const opts = vm.$options
  if (opts.props) initProps(vm, opts.props)
  if (opts.methods) initMethods(vm, opts.methods)
  if (opts.data) {
    initData(vm)
  } else {
    observe(vm._data = {}, true /* asRootData */)
  }
  if (opts.computed) initComputed(vm, opts.computed)
  if (opts.watch && opts.watch !== nativeWatch) {
    initWatch(vm, opts.watch)
  }
}
```

initProps

```
function initProps (vm: Component, propsOptions: Object) {
  const propsData = vm.$options.propsData || {}
  const props = vm._props = {}
  // cache prop keys so that future props updates can iterate using Array
  // instead of dynamic object key enumeration.
  const keys = vm.$options._propKeys = []
  const isRoot = !vm.$parent
  // root instance props should be converted
  if (!isRoot) {
    toggleObserving(false)
  }
  for (const key in propsOptions) {
    keys.push(key)
    const value = validateProp(key, propsOptions, propsData, vm)
    /* istanbul ignore else */
    if (process.env.NODE_ENV !== 'production') {…
    } else {
      defineReactive(props, key, value)
    }
    // static props are already proxied on the component's prototype
    // during Vue.extend(). We only need to proxy props defined at
    // instantiation here.
    if (!(key in vm)) {
      proxy(vm, `_props`, key)
    }
  }
  toggleObserving(true)
}
```

method和watch

```
function initMethods (vm: Component, methods: Object) {
  const props = vm.$options.props
  for (const key in methods) {
    if (process.env.NODE_ENV !== 'production') {…
    }
    vm[key] = typeof methods[key] !== 'function' ? noop : bind(methods[key], vm)
  }
}

function initWatch (vm: Component, watch: Object) {
  for (const key in watch) {
    const handler = watch[key]
    if (Array.isArray(handler)) {
      for (let i = 0; i < handler.length; i++) {
        createWatcher(vm, key, handler[i])
      }
    } else {
      createWatcher(vm, key, handler)
    }
  }
}
```

initData

```
function initData (vm: Component) {
  let data = vm.$options.data
  data = vm._data = typeof data === 'function'
    ? getData(data, vm)
    : data || {}
  if (!isPlainObject(data)) {
    data = {}
    process.env.NODE_ENV !== 'production' && warn(
      'data functions should return an object:\n' +
      'https://vuejs.org/v2/guide/components.html#data-Must-Be-a-Function',
      vm
    )
  }
  // proxy data on instance
  const keys = Object.keys(data)
  const props = vm.$options.props
  const methods = vm.$options.methods
  let i = keys.length
  while (i--) {
    const key = keys[i]
    if (process.env.NODE_ENV !== 'production') {…
    }
    if (props && hasOwn(props, key)) {…
    } else if (!isReserved(key)) {
      proxy(vm, `_data`, key)
    }
  }
  // observe data
  observe(data, true /* asRootData */)
}
```

响应式

```
 */
export function observe (value: any, asRootData: ?boolean): Observer | void {
  if (!isObject(value) || value instanceof VNode) {
    return
  }
  let ob: Observer | void
  if (hasOwn(value, '__ob__') && value.__ob__ instanceof Observer) {
    ob = value.__ob__
  } else if (
    shouldObserve &&
    !isServerRendering() &&
    (Array.isArray(value) || isPlainObject(value)) &&
    Object.isExtensible(value) &&
    !value._isVue
  ) {
    ob = new Observer(value)
  }
  if (asRootData && ob) {
    ob.vmCount++
  }
  return ob
}
```

```
export function defineReactive (
  obj: Object,
  key: string,
  val: any,
  customSetter?: ?Function,
  shallow?: boolean
) {
  const dep = new Dep()
  Object.defineProperty(obj, key, {
    enumerable: true,
    configurable: true,
    get: function reactiveGetter () {
      const value = getter ? getter.call(obj) : val
      if (Dep.target) {
        dep.depend()
        if (childOb) {
          childOb.dep.depend()
          if (Array.isArray(value)) {
            dependArray(value)
          }
        }
      }
      return value
    },
    set: function reactiveSetter (newVal) {
      const value = getter ? getter.call(obj) : val
      /* eslint-disable no-self-compare */
      if (newVal === value || (newVal !== newVal && value !== value)) {
        return
      }
      /* eslint-enable no-self-compare */
      if (process.env.NODE_ENV !== 'production' && customSetter) {
```

$mount

```
6
7   const mount = Vue.prototype.$mount
8   Vue.prototype.$mount = function (
9     el?: string | Element,
0     hydrating?: boolean
1   ): Component {
2     el = el && query(el)
3
4     /* istanbul ignore if */
5     if (el === document.body || el === document.documentElement) {
6       process.env.NODE_ENV !== 'production' && warn(
7         `Do not mount Vue to <html> or <body> - mount to normal elements instead.`
8       )
9       return this
0     }
1
2     const options = this.$options
3     // resolve template/el and convert to render function
4     if (!options.render) {
5       let template = options.template
6       if (template) {
7         if (typeof template === 'string') {
8           if (template.charAt(0) === '#') {
9             template = idToTemplate(template)
0             /* istanbul ignore if */
1             if (process.env.NODE_ENV !== 'production' && !template) {
2               warn(
3                 `Template element not found or is empty: ${options.template}`,
4                 this
5               )
6             }
```

```
    }
    // 只写了template，调用编译模块，吧template解析成render函数，返回虚拟dom
    const { render, staticRenderFns } = compileToFunctions(template, {
      outputSourceRange: process.env.NODE_ENV !== 'production',
      shouldDecodeNewlines,
      shouldDecodeNewlinesForHref,
      delimiters: options.delimiters,
      comments: options.comments
    }, this)
    options.render = render
    options.staticRenderFns = staticRenderFns

    /* istanbul ignore if */
    if (process.env.NODE_ENV !== 'production' && config.performance && mark) {
      mark('compile end')
      measure(`vue ${this._name} compile`, 'compile', 'compile end')
    }
  }
  return mount.call(this, el, hydrating)
}
```

```ts
// public mount method
Vue.prototype.$mount = function (
  el?: string | Element,
  hydrating?: boolean
): Component {
  el = el && inBrowser ? query(el) : undefined
  return mountComponent(this, el, hydrating)
}
```

mountComponent 完毕

```
export function mountComponent (
  vm: Component,
  el: ?Element,
  hydrating?: boolean
): Component {
  vm.$el = el
  if (!vm.$options.render) {
    vm.$options.render = createEmptyVNode
    if (process.env.NODE_ENV !== 'production') {…
    }
  }
  callHook(vm, 'beforeMount')

  let updateComponent
  /* istanbul ignore if */
  if (process.env.NODE_ENV !== 'production' && config.performance && mark) {…
  } else {
    updateComponent = () => {
      vm._update(vm._render(), hydrating)
    }
  }

  // we set this to vm._watcher inside the watcher's constructor
  // since the watcher's initial patch may call $forceUpdate (e.g. inside child
  // component's mounted hook), which relies on vm._watcher being already defined
  new Watcher(vm, updateComponent, noop, {
    before () {
      if (vm._isMounted && !vm._isDestroyed) {
        callHook(vm, 'beforeUpdate')
      }
    }
```

核心问题

- 响应式怎么收集依赖的
- template咋变成render的
- _update是啥
- _render是啥
- New Watcher是啥

组件化

- createElement
- 一切皆是vdom
- 用js的对象，来描述dom和组件
- 难点就是diffChildren
- 代码看起来

Vue3

- Composition 借鉴了react hooks
- 类似hooks，函数至上
- 逻辑组合和复用
- 完美支持ts
- tree-shaking
- 和hooks的区别 reactive vs diff