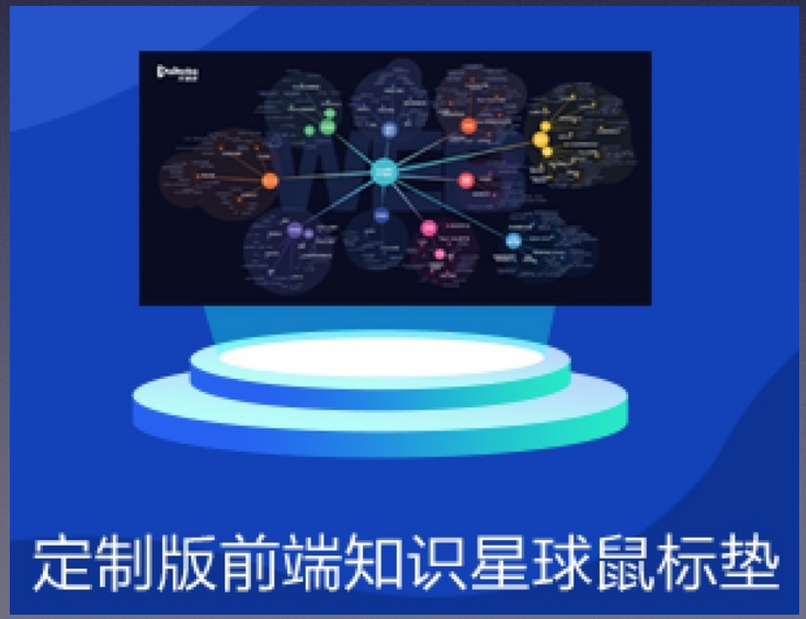
Vue 3.0训练营

村长@每晚8点

课前福利

口号：我要跟村长学习

课后福利



上节课回顾

今日目标

- 搞清楚数据响应式概念
- 响应式革新：vue2 vs vue3
- 手写实现
- vue3响应式模块源码剖析

什么是数据响应式

数据变化可侦测，和数据相关的内容可以更新。

```
// vue2
function defineReactive(obj, key, val) {
  Object.defineProperty(obj, key, {
    get() {
      return val
    },
    set(v) {
      val = v
      update()
    }
  })
}
```

vue3利用Proxy实现

```
function defineReactive(obj) {
  return new Proxy(obj, {
    get(target, key) {
      return target[key]
    },
    set(target, key, val) {
      target[key] = val
      update()
    }
  })
}
```

vue2 vs vue3

vue2需要遍历对象所有key，这会影响初始化速度。

```
function observe(obj) {
  if (typeof obj !== 'object' || obj == null) {
    return
  }

  const keys = Object.keys(obj)
  for (let i = 0; i < keys.length; i++) {
    const key = keys[i]
    defineReactive(obj, key, obj[key])
  }
}
```

vue2 vs vue3

vue2对于数组要做特殊处理，修改数据时也不能使用索引方式。

```
const originalProto = Array.prototype
const arrayProto = Object.create(originalProto)
;['push', 'pop', 'shift', 'unshift', 'splice', 'reverse', 'sort'].forEach(
  method => {
    arrayProto[method] = function() {
      originalProto[method].apply(this, arguments)
      dep.notify()
    }
  }
)
```

vue2 vs vue3

vue2中动态添加或删除对象属性需要使用额外API：Vue.set()/delete()

```
Vue.set(obj, 'bar', 'barrrrrrr')
Vue.delete(obj, 'bar')
```

vue2 vs vue3

vue3中利用Proxy可以很好的解决以上的问题

```
function reactive(obj) {
  if (typeof obj !== 'object' && obj != null) {
    return obj
  }
  // Proxy相当于在对象外层加拦截
  // http://es6.ruanyifeng.com/#docs/proxy
  const observed = new Proxy(obj, {
    get(target, key, receiver) {
      // Reflect用于执行对象默认操作，更规范、更友好
      // Proxy和Object的方法Reflect都有对应
      // http://es6.ruanyifeng.com/#docs/reflect
      const res = Reflect.get(target, key, receiver)
      console.log(`获取${key}:${res}`)
      return res
    },
    set(target, key, value, receiver) {
      const res = Reflect.set(target, key, value, receiver)
      console.log(`设置${key}:${value}`)
      return res
    },
    deleteProperty(target, key) {
      const res = Reflect.deleteProperty(target, key)
      console.log(`删除${key}:${res}`)
```

造个轮子

首先实现reactive(obj)，借助Proxy代理传入的obj，
这样可以拦截对obj的各种访问。

```
const baseHandler = {
  get(target, key, receiver) {},
  set(target, key, value, receiver) {},
  deleteProperty(target, key) {}
}

function reactive(obj) {
  return new Proxy(obj, baseHandler)
}
```

造个轮子

不要忘了对嵌套对象的处理

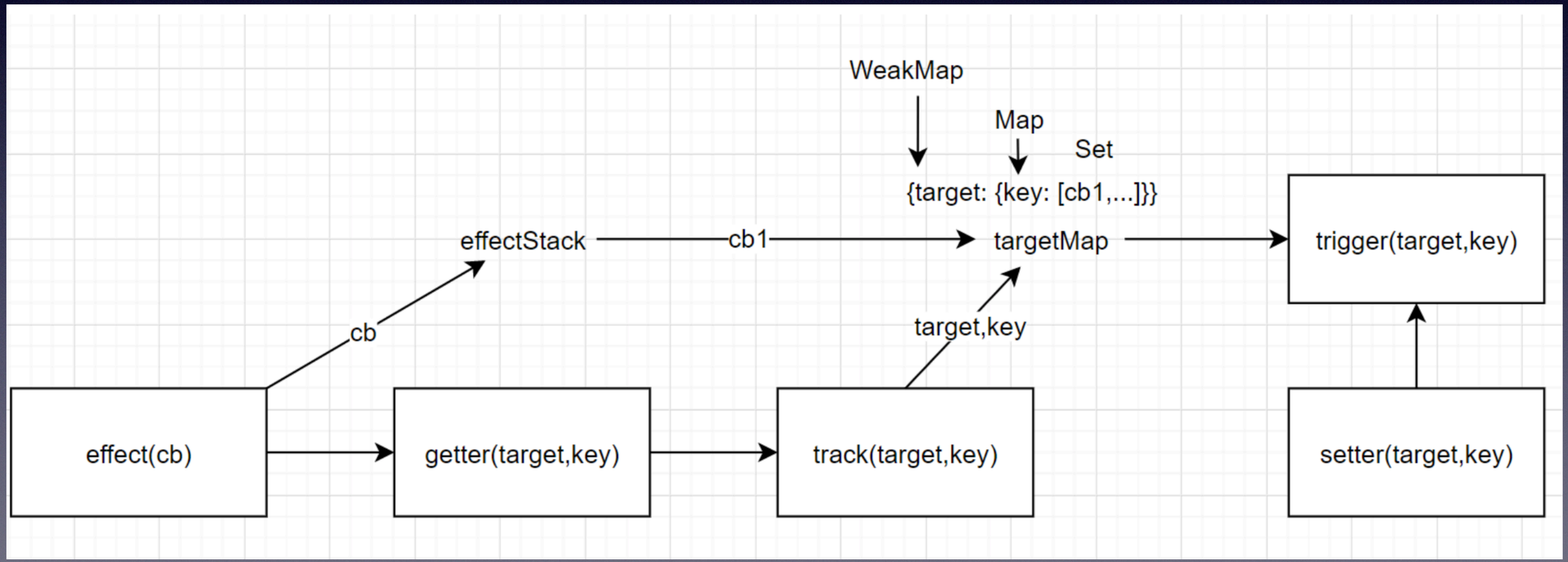
```
// 传入对象应该是一个非null的object
const isObject = v => typeof v === 'object' && v !== null

const baseHandler = {
  get(target, key, receiver) {
    const res = Reflect.get(target, key, receiver)
    // 判断res是对象，递归处理它
    return isObject(res) ? reactive(res) : res
  },
}
function reactive(obj) {
  // reactive()只接受非null的object
  if (!isObject(obj)) {
    return obj
  }
}
```

小姐姐时间

造个轮子

最后一步是依赖收集，整体思路如下



造个轮子

代码基本结构

```
// 临时存储响应式函数
const effectStack = []

// 将传入fn转换为一个响应式函数
function effect(fn, options = {}) {}

// 存放响应式函数和目标、键之间的映射关系
const targetMap = new WeakMap()

// 依赖收集，创建映射关系
function track(target, key) {}

// 根据映射关系获取响应函数
function trigger(target, key) {}
```

总结时间

- 内容总结
- 今天我学到了啥?

送福利环节

下次课内容预告

- 明天6~7点直播答疑、交流

课后福利

口号：明天继续唠