1. 知识要点都是要掌握的吗?
    1. 基础一定要掌握
        1. js运行机制
        2. ES6
        3. 没有这个基础，很难做源码分析
    2. 通用的技能，有一个深入 掌握的即可
        1. vue，react，node, 工程化，小程序
        2. 每一个都是入门，其实没意义
        3. 一定有一个源码级的技能
        4. 今天带着大家回顾一下vue源码
2. vue源码学习细节
    1. vue2引入的虚拟dom 为什么引入
        1. 我们有了watcher，每个变量变化，都知道，能直接更新
            1. 直接就知道了diff的结果
        2. 虚拟dom是通过diff 来算出哪里变了，再更新dom 依然是要更新dom，只不过让更新最少
        3. 为啥需要虚拟dom
            1. 减少dom操作的次数
            2. 虚拟dom就是用js对象，描述dom结构
        4. vue1的时候，每个组件恩{{}}都有一个watcher，vue2一个组件只有一个watcher， 组件内部的众多变量修改，只通知到组件，组件内部虚拟dom diff来算出修改的变量
    2. Object。definproperty的缺点
        1. 数组监听不到
        2. vue拦截数组常用方法，通知更新
    3. vue的虚拟dom有啥特点
    4. vue源码执行的流程
        1. 刚才分析的过程 整齐启动的过程
    5. vue.extend是啥
    6. vue.use是啥
    7. 类似的问题，可以问100个 只要看了源码 ，不变应万变 不用刷题
    8. 工作是一辈子的事， 不要突击

不在模板里面用到的变量，是不是不放data里比较好，因为放了就要有watcher监听

是

1. 找入口

    1. import Vue from 'vue'
    2. vue这个项目 package.json种，*module*的字段
2. core/instance/index
3. 入口执行了this._init
4. initMixin 扩展函数
    1. 扩展了_init
    2. initLifecycle
        1. 修正父元素的$child
        2. 初始化refs children $parent
        3.
    3. initEvents
        1. 初始化vm._events
    4. initRender
        1. vm._c 就是createElement ， 在compile模块，会用到 `___c`
        2. vm.$createElement 也是createElement （重点看的函数）
        3. $attrs$和 Listeners
    5. initInjections
        1. 定了向上查找provide的逻辑 vm._provided
    6. initState data 响应式 都在这 （重要）
        1. 初始化_watcher数组
        2. initProps props配置初始化
        3. initMethods
            1. methods函数，挂载在vm之上 所以才可以直接通过this.获取到
        4. initData
        5. observe
        6. initComputed
            1. computed通过watcher存储在vm._computedWatchers里
            2. computed和watcher核心都是Watcher，但是多了缓存的控制
        7. initWatch
            1. 执行的是vm.$watch
    7. initProvide
        1. vm._provided
        2. provider可以是函数
    8. 如果有el配置，执行$mount
5. stateMixin
    1. $set
    2. $delete
    3. $watch
6. eventsMixin
    1. onOnce offemt 时间存储在 vm._events下面

7. lifecycleMixin
    1. _update 重要
        1. 数据更新
        2. 渲染 无论是首次，还是后续的更新，都是执行 **patch** （重点学习**patch**）
    2. forceUpdate
        1. 强制更新，执行的是vm._watcher.update()， watcher是啥
    3. destory 销毁
8. renderMixin
    1. $nextTick
        1. 执行的是nextTick （后续看）
    2. _render （重点学习渲染过程） 生成虚拟dom
        1. vnode = render.call(vm._renderProxy, vm.$createElement)
        2. 执行render函数，传入vm和$createElement

1. rutime-with-compileer
    1. $mount 修正
    2. runtime-index
        1. 定义**patch**
        2. 定义$mount
        3. core/index
            1. initGlobalAPI

```
Vue.util = {
  warn,
  extend,
  mergeOptions,
  defineReactive
}

Vue.set = set
Vue.delete = del
Vue.nextTick = nextTick
export const ASSET_TYPES = [
  'component',
  'directive',
  'filter'
]
{
    data(){

    }
```

```
    components:{},
    filters:{},
    directives:{}
}
  ASSET_TYPES.forEach(type => {
    Vue.options[type + 's'] = Object.create(null)
  })
注册keep-alive
initUse 初始化vue.use 插件机制
initMixin  Vue.mixin 合并配置
initExtend Vue.extend 继承机制
initAssetRegisters
```

3.

vue这么多版本，with-compileer是干啥的

看实际的vuejs源码 来体验一下，课堂写过迷你的，和实际的区别