Vue源码剖析01

目标

- 习
- vue初 剖
- 入

知识点

获取vue

：https://github.com/vuejs/vue

出 ：g c e h ://g h b.c / e / e.g

前 ：2.6.11

文件结构


VUE
.circleci
.github
.vscode
benchmarks
dist 发布目录
examples 范例，里面有测
flow 试代码
node_modules
packages 核心代码之外的独立库
scripts 构建脚本
src 源码
test
types ts类型声明，上面flow是针对flow的类型声明
.babelrc.js
.editorconfig
.eslintignore
.eslintrc.js
.flowconfig
.gitignore
BACKERS.md
LICENSE
package.json
README.md
yarn.lock

调试环境搭建

- 依 ：

  phantom.js

- rollup： -g
- 修 dev ， 加sourcemap，package.json

```
"de ": "      -  -c  c    /c  f g.   --    ce a  --e      e   TARGET: eb-
f   -de ",
```

- 令： de
- 入前 创 vue.js，samples/commits/index.html

```
< c      c="../../d   /  e.  "></ c    >
```

:

- runtime：仅包 ，不包
- common：cjs ， 于webpack1
- esm：ES ， 于webpack2+
- umd: universal module definition，兼 cjs amd， 于

入口

dev 中`-c c /c f g.` 件

`TARGET: eb-f -de` 出 件 ，line:123

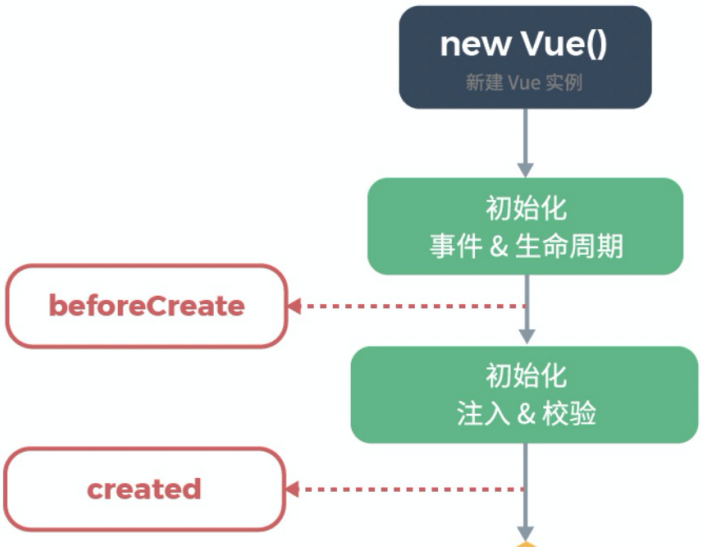
```
// R e+c e de e e b d (B e )

' eb-f -de ':
  e : e e(' eb/e - e- h-c e . '), // 入口
  de : e e('d / e. '),// 目标文件
  f a : ' d', // 输出规范
  e : 'de e e ',
  a a : he: './e -dec de ' ,
  ba e ,
 ,
```

初始化流程

整体流程

- new Vue()
  - _init()
- $mount()
  - mountComponent()
    - updateComponent()
      - render()
      - update()
    - new Watcher()



例：01-init.html

入口 platforms/web/entry-runtime-with-compiler.js

$mount : template el

platforms/web/runtime/index.js

web 令 件

义__patch__: 丁函 , patching

义$mount: vue 例到 主元 (dom 主元)

core/ ndex.js

初 全 api

具体 下:

```
V e. e  =  e
V e.de e e = de
V e. e  T c  =  e  T c
    U e(V e) // 实现V e.  e函数
    M    (V e) // 实现V e.     函数
    E  e d(V e) // 实现V e.e  e d函数
    A  e Reg   e  (V e) // 注册实现V e.c     e  /d  ec   e/f   e
```

core/instance/index.js

Vue 函 义

义Vue 例API

```
f  c     V e (       )
  // 构造函数仅执行了_
   h  ._    (       )


    M    (V e)  // 实现     函数
  a eM    (V e) // 状态相关a   $da a,$     ,$ e ,$de e e,$ a ch
e e   M    (V e)// 事件相关a   $  ,$  ce,$ ff,$e
  fec c eM    (V e) // 生命周期a   _  da e,$f  ceU da e,$de
 e de M    (V e)// 渲染a   _ e de ,$ e  T c
```

core/instance/init.js

创 件 例，初 其 事件

```
    L fec c e(  ) // $ a e  ,$    ,$ch  d e ,$ ef
    E e   (  )    // 处理父组件传递的事件和回调
    Re de (  )    // $     ,$ c  edS    ,_c,$c ea eE e e
ca  H   (  , 'bef  eC ea e')
    I  ec     (  ) // 获取注入数据
    S a e(  )      // 初始化     ，  e h d ，da a，c     ed，  a ch
    P    de(  )    // 提供数据注入
ca  H   (  , 'c ea ed')
```

$mount

- mountComponent

  ， vdom 为dom

- new Watcher()

创 件 watcher

- updateComponent()

初

- update()

初 ， 传入vdom 为dom，初 dom创 作

- render() src\core\instance\render.js

  件， vdom

  代 ：examples\test\01-init.html

整体流程捋一捋

new Vue() => _init() => $mount() => mountComponent() =>

new Watcher() => updateComponent() => render() => _update()

> 关 : vue
>
> - : 件创
> - : 使 们 以 做 事
> - 分 列举:
>   - 初 : beforeCreate created beforeMount mounted
>   - : beforeUpdate updated
>   - : beforeDestroy destroyed
> - :
>   - created , 准 , 做 值 作
>   - mounted , $el , 以 dom; 件也 , 以 们
>   - updated , 值 作 于dom, 以 dom
>   - destroyed , 件 例 , 作

数据响应式

MVVM , 以 Vue中利 了JS Object.defineProperty(), 义 getter/setter

具体 Vue初 , 会 initState, 会初 data，props , 关 data初 ,

整体流程

initState (vm: Component) src\core\instance\state.js

初 , 包 props methods data computed watch

initData核心代码是将data数据响应化

```
f  c        Da a (  : C     e  )
  // 执行数据响应化
   b e  e(da a,    e /* a R   Da a */)
```

core/observer/index.js

observe 个Observer 例

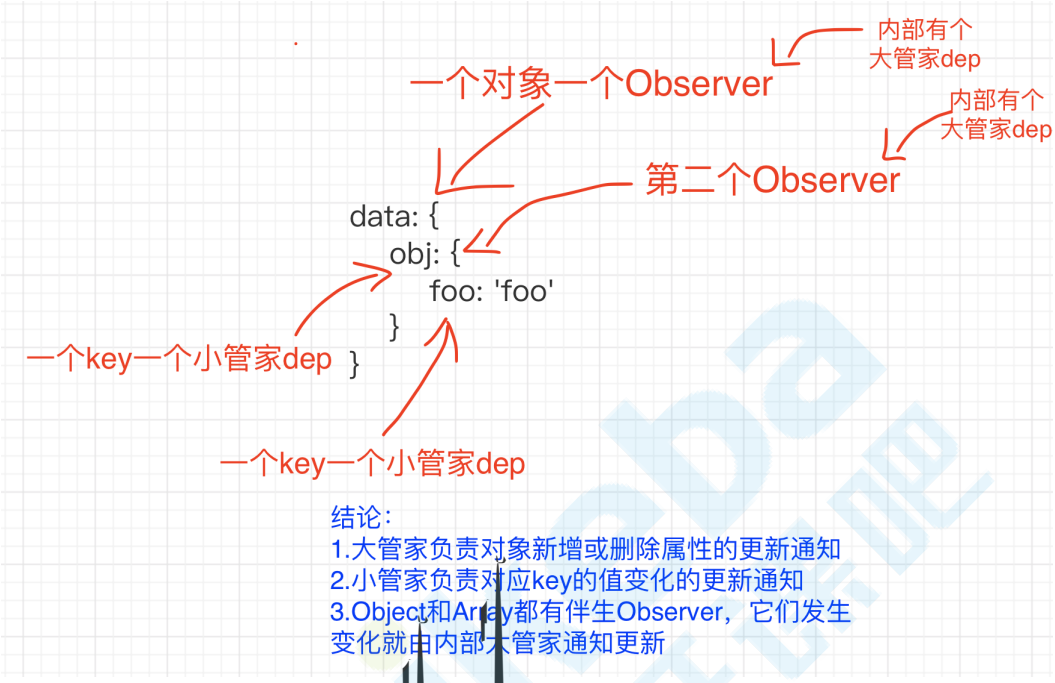
core/observer/index.js

Observer 作

defineReactive 义 getter/setter，getter 加依 ，setter

core/observer/dep.js

Dep Watcher，包 watcher 例 删



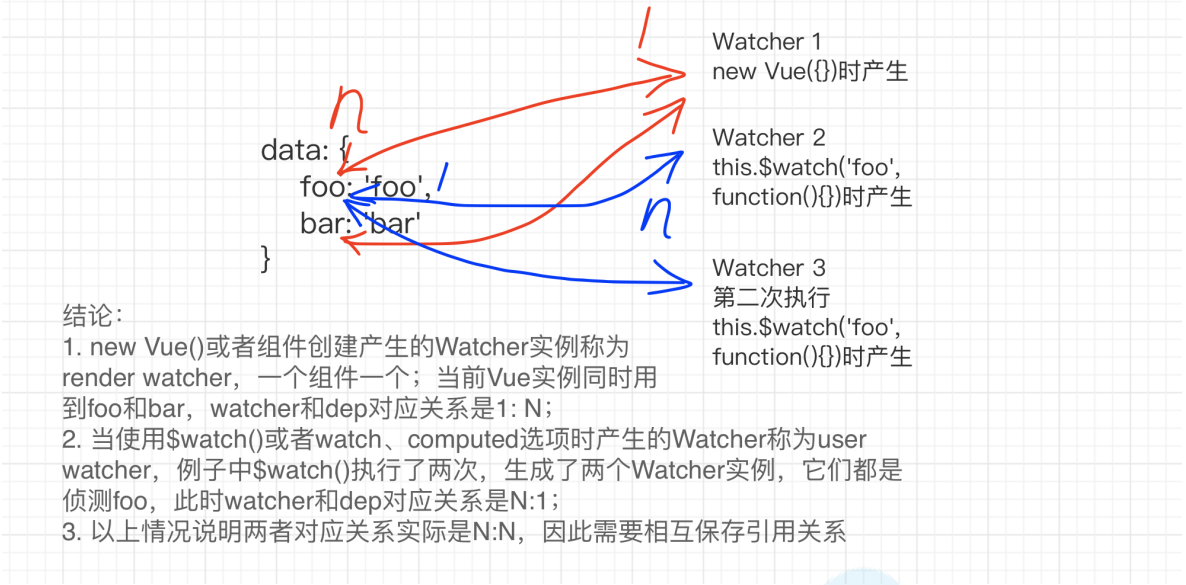
Watcher

Watcher 个 依 ， 值 函 ， 于$watch API 令中

个 件也会 Watcher， 值 会 其update函

```
e      defa    c a   Wa che
    c      c    ()
    ge  ()
    addDe  (de : De )
      da e ()
```

> 关API：`$ a che`



结论：
1. new Vue()或者组件创建产生的Watcher实例称为render watcher，一个组件一个；当前Vue实例同时用到foo和bar，watcher和dep对应关系是1: N；
2. 当使用$watch()或者watch、computed选项时产生的Watcher称为user watcher，例子中$watch()执行了两次，生成了两个Watcher实例，它们都是侦测foo，此时watcher和dep对应关系是N:1；
3. 以上情况说明两者对应关系实际是N:N，因此需要相互保存引用关系

代 examples\test\02-1-reactive.html

数组响应化

侦 不 ， 们 作 使 push pop splice ， 办
以vue中 些 dep

src\core\observer\array.js

为 中 7个 以 内 义

Observer中覆盖数组原型

```
f (A  a .  A  a ( a  e))
  // 替换数组原型
      A g e  ( a  e, a  a Me h d ) //  a  e.__     __ = a  a Me h d
   h  . b e  eA  a ( a  e)
```

代 examples\test\02-2-reactive-arr.html

> 关API：`V e. e ()/de e e()`

```
da a:
  a  : []
```

```
a  . e g h = 0
a  [  de ] =

V e. e ()
V e.de ()
```

作业:初始化流程思维导图

-
-
-