Vue version:2.5.13

**匿名自执行函数初始化Vue**

1-4 ( function (global,factory){

**typeof** exports === **'object'** && **typeof** module !== **'undefined'** ? module.**exports** = factory() :  
 **typeof** define === **'function'** && define.amd ? define(factory) :  
 (global.**Vue** = factory()); factory() =》 Vue$3 == Vue

} ( this,function(){}) ) )

*当我们引入Vue文件时，返回的是Vue构造函数 调用方法 new Vue() =>执行Vue$3 构造函数*

**4640** *定义构造函数*

**function** *Vue$3* (options) {  
 **console**.log(**this**.**constructor**.**options**)  
 **if** (**"development"** !== **'production'** &&  
 !(**this instanceof** *Vue$3*)  
 ) {  
 warn(**'Vue is a constructor and should be called with the `new` keyword'**);  
 }  
 **this**.\_init(options);  
}

**在处理传递的参数【options】之前，Vue进行了一系列的参数处理**

**4983** *initGlobalAPI*(*Vue$3*);

**4939function** *initGlobalAPI* (Vue) {  
 *// config* **var** configDef = {};  
 configDef.get = **function** () { **return** config; };  
 { 形成局部作用域，按顺序执行  
 configDef.set = **function** () {  
 warn(  
 **'Do not replace the Vue.config object, set individual fields instead.'** );  
 };  
 }  
 Object.defineProperty(Vue, **'config'**, configDef);

网址讲解

*//https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global\_Objects/Object/defineProperty* *// exposed util methods.  
 // NOTE: these are not considered part of the public API - avoid relying on  
 // them unless you are aware of the risk.*

Vue.**util** = {  
 warn: warn,  
 extend: *extend*,  
 mergeOptions: *mergeOptions*,  
 defineReactive: *defineReactive* };  
  
 Vue.set = *set*;  
 Vue.delete = *del*;  
 Vue.nextTick = *nextTick*;  
  
 Vue.**options** = Object.create(**null**);  
 ASSET\_TYPES.forEach(**function** (type) {  
 Vue.**options**[type + **'s'**] = Object.create(**null**);  
 });

**230** *// var ASSET\_TYPES = ['component','directive','filter'];*

*// this is used to identify the "base" constructor to extend all plain-object  
 // components with in Weex's multi-instance scenarios.* Vue.**options**.**\_base** = Vue;  
  
 *extend*(Vue.**options**.**components**, builtInComponents);  
  
 *initUse*(Vue);  
 *initMixin$1*(Vue);  
 *initExtend*(Vue);  
 *initAssetRegisters*(Vue);  
}

4933

接下来执行依次执行以下内容

**215** *extend*(Vue.**options**.**components**,builtInComponents);

**4652** *initUse*(Vue);

**4674** *initMixin$1*(Vue);  
**4683** *initExtend*(Vue);

**4775** *initAssetRegisters*(Vue);

*extend*(Vue.**options**.**components**,builtInComponents);

**function** *extend* (to, from) {  
 **for** (**var** key **in** from) {  
 to[key] = from[key];  
 }  
 **return** to  
}

Vue.**options**.**components ==》 { }**

**builtInComponents ==>** {**KeepAlive**: KeepAlive};

**var KeepAlive** = {  
 **name**: **'keep-alive'**,  
 **abstract**: **true**,  
  
 **props**: {  
 **include**: patternTypes,  
 **exclude**: patternTypes,  
 **max**: [String, Number]  
 },  
  
 created: **function** created () {  
 **this**.cache = Object.create(**null**);  
 **this**.**keys** = [];  
 },  
  
 destroyed: **function** destroyed () {  
 **var** this$1 = **this**;  
  
 **for** (**var** key **in** this$1.**cache**) {  
 *pruneCacheEntry*(this$1.**cache**, key, this$1.**keys**);  
 }  
 },  
  
 **watch**: {  
 include: **function** include (val) {  
 *pruneCache*(**this**, **function** (name) { **return** *matches*(val, name); });  
 },  
 exclude: **function** exclude (val) {  
 *pruneCache*(**this**, **function** (name) { **return** !*matches*(val, name); });  
 }  
 },  
  
 render: **function** render () {  
 **var** slot = **this**.**$slots**.**default**;  
 **var** vnode = *getFirstComponentChild*(slot);  
 **var** componentOptions = vnode && vnode.**componentOptions**;  
 **if** (componentOptions) {  
 *// check pattern* **var** name = *getComponentName*(componentOptions);  
 **var** ref = **this**;  
 **var** include = ref.include;  
 **var** exclude = ref.**exclude**;  
 **if** (  
 *// not included* (include && (!name || !*matches*(include, name))) ||  
 *// excluded* (exclude && name && *matches*(exclude, name))  
 ) {  
 **return** vnode  
 }  
  
 **var** ref$1 = **this**;  
 **var** cache = ref$1.**cache**;  
 **var** keys = ref$1.**keys**;  
 **var** key = vnode.**key** == **null** *// same constructor may get registered as different local components  
 // so cid alone is not enough (#3269)* ? componentOptions.**Ctor**.**cid** + (componentOptions.**tag** ? (**"::"** + (componentOptions.**tag**)) : **''**)  
 : vnode.**key**;  
 **if** (cache[key]) {  
 vnode.**componentInstance** = cache[key].**componentInstance**;  
 *// make current key freshest  
 remove*(keys, key);  
 keys.push(key);  
 } **else** {  
 cache[key] = vnode;  
 keys.push(key);  
 *// prune oldest entry* **if** (**this**.**max** && keys.**length** > parseInt(**this**.**max**)) {  
 *pruneCacheEntry*(cache, keys[0], keys, **this**.**\_vnode**);  
 }  
 }  
  
 vnode.**data**.**keepAlive** = **true**;  
 }  
 **return** vnode || (slot && slot[0])  
 }  
};