Vue.js 依赖收集

作者：answershuto
原文链接：https://ld246.com/article/1503885492616
来源网站：链滴

写在前面

因为对Vue.js很感兴趣，而且平时工作的技术栈也是Vue.js，这几个月花了些时间研究学习了一下Vue.s源码，并做了总结与输出。

文章的原地址：https://github.com/answershuto/learnVue。

在学习过程中，为Vue加上了中文的注释https://github.com/answershuto/learnVue/tree/master/ue-src，希望可以对其他想学习Vue源码的小伙伴有所帮助。

可能会有理解存在偏差的地方，欢迎提issue指出，共同学习，共同进步。

依赖收集是在响应式基础上进行的，不熟悉的同学可以先了解《响应式原理》。

为什么要依赖收集

先看下面这段代码

```
new  Vue({

template:

`

text1: {{text1}}

text2: {{text2}}

`,

data: {

text1:  'text1',

text2:  'text2',

text3:  'text3'

}

});
```

按照之前《响应式原理》中的方法进行绑定则会出现一个问题——text3在实际模板中并没有被用到然而当text3的数据被修改的时候（this.text3 = 'test'）的时候，同样会触发text3的setter导致重新行渲染，这显然不正确。

先说说Dep

当对data上的对象进行修改值的时候会触发它的setter，那么取值的时候自然就会触发getter事件，以我们只要在最开始进行一次render，那么所有被渲染所依赖的data中的数据就会被getter收集到De的subs中去。在对data中的数据进行修改的时候setter只会触发Dep的subs的函数。

定义一个依赖收集类Dep。

```
class  Dep () {
constructor () {
this.subs  = [];
}
addSub (sub:  Watcher) {
this.subs.push(sub)
}
removeSub (sub:  Watcher) {
remove(this.subs, sub)
}
notify () {
// stabilize the subscriber list first
const  subs  =  this.subs.slice()
for (let  i  =  0, l  =  subs.length; i  <  l; i++) {
subs[i].update()
}
}
}
```

Watcher

订阅者，当依赖收集的时候回addSub到sub中，在修改data中数据的时候会触发Watcher的notify，而回调渲染函数。

```
class  Watcher () {
constructor (vm, expOrFn, cb, options) {
this.cb  =  cb;
this.vm  =  vm;
/在这里将观察者本身赋值给全局的target，只有被target标记过的才会进行依赖收集/
Dep.target  =  this;
/触发渲染操作进行依赖收集/
```

```
this.cb.call(this.vm);
}
update () {
this.cb.call(this.vm);
}
}
```

开始依赖收集

```
class Vue {
constructor(options) {
this._data = options.data;
observer(this._data, options.render);
let watcher = new Watcher(this, );
}
}
function defineReactive (obj, key, val, cb) {
/ 在闭包内存储一个Dep对象 /
const dep = new Dep();
Object.defineProperty(obj, key, {
enumerable: true,
configurable: true,
get: ()=>{
if (Dep.target) {
/Watcher对象存在全局的Dep.target中/
dep.addSub(Dep.target);
}
},
set:newVal=> {
/只有之前addSub中的函数才会触发/
```

```
dep.notify();

}

})

}

Dep.target = null;
```

将观察者Watcher实例赋值给全局的Dep.target，然后触发render操作只有被Dep.target标记过的才进行依赖收集 。有Dep.target的对象会讲Watcher的实例push到subs中，在对象被修改出发setter作的时候dep会调用subs中的Watcher实例的update方法进行渲染。