vue双向数据绑定简单实现

vue双向数据绑定(一).mp4

1. getSet方法

定义一个对象,让它的属性,可以分成set和get两个方法来获取或者设置它的值

> person.js

```
var  person  = {
  _name:null,
  get name(){
    return this._name
  },
  set name(val) {
    this._name = val
  }
}
person.name = 'Wu.Yu'
console.log(person.name)
```

2.发布订阅模式

demo-1

简单实现发布订阅模式

1. 在构造函数中,定义一个数组.用于存放事件订阅者.
2. 在addListener函数中,把事件推送到队列中,
3. trigger中,循环订阅者数组,如果传进来的是函数,直接使用**call(this)**,执行该函数

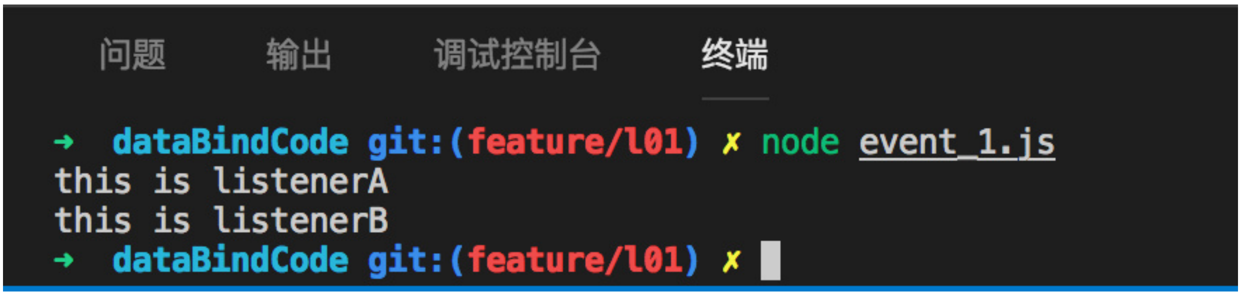
> event_1.js

```
class Event {
  constructor(){
    this.listeners = []
  }
  addListener(listener){
    this.listeners.push(listener)
  }
  trigger(){
    for (let i = 0; i < this.listeners.length; i++) {
      this.listeners[i].call(this)
    }
  }
}

var myEvents = new Event();
myEvents.addListener(function (params) {
  console.log("this is listenerA");

})
myEvents.addListener(function (params) {
  console.log("this is listenerB");
})

myEvents.trigger();
```

调试 event.js函数

```
node event_1.js
```

截图如下所示:



```
➜  dataBindCode git:(feature/l01) ✗ node event_1.js
this is listenerA
this is listenerB
➜  dataBindCode git:(feature/l01) ✗ 
```

demo-2

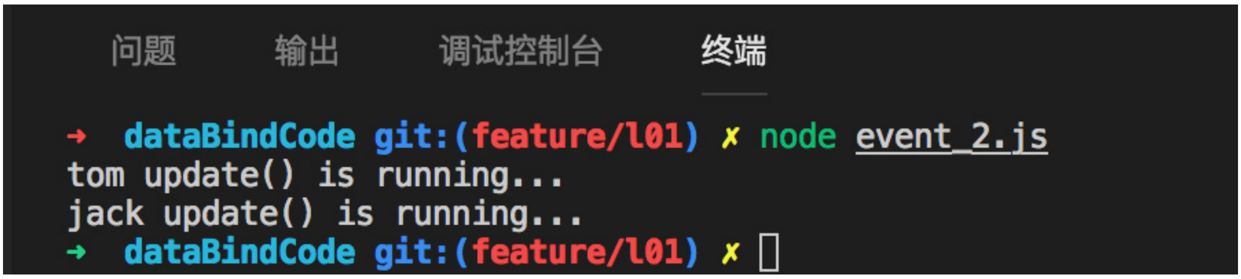
> event_2.js

```
class Event {
  constructor(){
    this.listeners = []
  }
  addListener(listener){
    this.listeners.push(listener)
  }
  trigger(){
    for (let i = 0; i < this.listeners.length; i++) {
      this.listeners[i].update()
    }
  }
}

var myEvents = new Event();
var tom = {
  update:function() {
    console.log("tom update() is running...");

  }
}

var jack = {
  update:function() {
    console.log("jack update() is running...");

  }
}

myEvents.addListener(tom)
myEvents.addListener(jack)
myEvents.trigger();
```

调试 event.js函数

```
node event_2.js
```

截图如下所示:



```
➜  dataBindCode git:(feature/l01) ✗ node event_2.js
tom update() is running...
jack update() is running...
➜  dataBindCode git:(feature/l01) ✗ 
```