面向对象20170515

自定义滚轮事件

```
addEvent(document,"上滚", function () {
   alert(1)
})
addEvent(document,"xia滚", function () {
   alert(xia)
})

function addEvent(obj,eventName,fn){//映射器（同一事件绑定多个函数）
    obj.zdy = obj.zdy || {};
       obj.zdy[eventName] = obj.zdy[eventName] || [ ];
     obj.zdy[eventName].push (fn)

}
function trigger(obj,events){//触发器（循环调用函数）
   if(!obj.zdy[events])return;
   obj.zdy[events].forEach((e,i)) =>{
      e();

   }
}
document.onmousewheel = function(ev){
   if(ev.wheelDelta>0){
      trigger(document,'上滚');
   }else{
      trigger(document,'下滚')
   }
}
```

//用面向对象写封装组件

1. 封装组件的原则（步骤）
   写组件的时候不要把容易修改的部分写死（最好留回调或者配置参数）
2. 自定义事件

```
        function Drag(id) {
            this.box = document.getElementById(id);
            this.disX = 0;
            this.disY = 0;
            const _this = this;
            this.settings = {
                cdown: function() {
                    _this.box.style.background = 'yellow';
                },
                cmove: function() {
                    _this.box.style.background = 'green';
                },
                cup: function() {
                    _this.box.style.background = 'red';
        }
    }
}

Drag.prototype.init = function(json) {
    for (var attr in json) {
        if (this.settings[attr] && typeof this.settings[attr] == typeof json[attr]) {
            this.settings[attr] = json[attr];
        }
    }
    const _this = this; //实例化对象
    this.box.addEventListener('mousedown', function(ev) {
        //this 元素
        _this.down(ev);
    });
}

Drag.prototype.down = function(ev) {
    this.settings.cdown();
    this.trigger('按下');
    this.disX = ev.pageX - this.box.offsetLeft;
    this.disY = ev.pageY - this.box.offsetTop;

    const _this = this;

    const fnMove = function(ev) {
        _this.move(ev);
    }
    const fnUp = function(ev) {
        _this.up(ev, fnMove, fnUp);
    }

    document.addEventListener('mousemove', fnMove); //this.move
    document.addEventListener('mouseup', fnUp);

    ev.preventDefault();
}

Drag.prototype.move = function(ev) {
    this.settings.cmove();
    this.box.style.left = ev.pageX - this.disX + 'px';
    this.box.style.top = ev.pageY - this.disY + 'px';
}

Drag.prototype.up = function(ev, move, up) {
    this.settings.cup();
    document.removeEventListener('mousemove', move);
    document.removeEventListener('mouseup', up);
}7
    1.绑定函数（把相同事件的不同函数push到一个数组中）
    2.触发（当指定事件调用之后,循环数组中的每个函数，并且调用）
Drag.prototype.addEventListener = function(events, fn) {
    this.zdy = this.zdy || {};
    this.zdy[events] = this.zdy[events] || [];
    this.zdy[events].push(fn);
}

Drag.prototype.trigger = function(events) {
    if (!this.zdy[events]) return;

    this.zdy[events].forEach((e, i) => {
        //          console.log(this);
        e.call(this);
    });
}
var d = new Drag('div');
//A同学开发的  5年
d.init({
    cup: function() {
        d.box.style.background = 'pink';
    },
    cdown: function() {
        d.box.style.background = 'blue';
    },
    opt: true
});
d.addEventListener('按下', function() {
    this.box.style.border = '2px solid #000';
    console.log(this);
});
```