定时器以及JS中动画的详细剖析

定时器详细解读

> JS中的定时器有两种
> `setInterval`
> `setTimeout`

setInterval

> 设置一个定时器，等待[interval]这么长时间后，会把[function]执行，以后每间隔[interval]这么长的时间，都会重新的再次把[function]执行，直到手动的去清除定时器为止
> (`clearInterval/clearTimeout`)

`window.setInterval([function],[interval])`

setTimeout

> 设置一个定时器，等待[interval]这么长的时间后，执行[function]，执行完成当前定时器停止工作(不是定时器清除,而是不工作了而已)，相对与setInterval来说，setTimeout执行一次就结束了

`window.setTimeout([function],[interval])`

```
var n = 0;
window.setInterval(function () {
    console.log(++n);//->输出多次,从1开始一直累加下去
}, 1000);

var m = 0;
window.setTimeout(function () {
    console.log(++m);//->输出一次1
}, 1000);
```

定时器中的THIS和实参

> 真实项目中，我们往往不仅想到达时间后执行一个函数，而且有时候需要改变这个函数中的THIS以及给函数传递一些实参，那么我们该如何处理呢？

```
1. //->定时器在语法上还支持第三个参数，第三个参数写的值，是在函数执行的时候，给函数传递的实参值(在IE9及以下版本的浏览器中不兼容)
2. window.setTimeout(function (str) {
3.     console.log(str);//->'i am parameter'
4. }, 1000, 'i am parameter');
5.
6. //->一般情况下,当时间到达,执行这个匿名函数的时候,函数中的THIS指向WINDOW,而且当前函数中没有传递任何的实参
7. window.setTimeout(function () {
8.     console.log(this === window);//->TRUE
9. }, 1000);
10.
11. var obj = {name: 'zhufeng'};
12. window.setTimeout(function (num) {
13.     //->使用CALL或者APPLY
14.     //在设置定时器的时候就把当前的匿名函数执行了,把函数执行的结果赋值给了定时器 <=> setTimeout(undefined,1000) 1000MS后执行的是UNDEFINED,不是我们想要的效果
15. }.call(obj, 100), 1000);
16.
17. window.setTimeout(function (num) {
18.     //->使用BIND(IE6~8下不兼容)
19.     //使用BIND可以解决这个问题,BIND仅仅是预先把函数中的THIS和参数都准备好,设置定时器的时候,第一个参数还依然是个函数,当1000MS后执行函数,此时函数中的THIS已经是预先设置的OBJ了,NUM的值也是预先设定好的100这个值
20. }.bind(obj, 100), 1000);
21.
22. //=>利用了闭包可以保存内容的机制:在设置定时器的时候,我们预先形成一个不销毁的闭包(把需要的THIS和NUM提前通过CALL改变了),在这函数中返回一个小函数给定时器,1000MS后执行返回的小函数,在小函数中我们可以通过作用域链的机制找到需要的OBJ以及NUM
23. var obj = {name: 'zhufeng'};
24. window.setTimeout(function (num) {
25.     //->this:obj num=100
26.     var _this = this;
27.     return function () {
```

```
        //->1000MS后执行的是返回的小函数,把需要处理的事情放在小函数中即可
        console.log(num);//->100
        console.log(this);//->window
        console.log(_this);//->obj
    }
}.call(obj, 100), 1000);
```

定时器的返回值

> 设置一个定时器，都会有返回值，返回值是一个数字，代表当前是第几个定时器，随着设置定时器的增多，当前这个数字会一直累加（类似于去银行办理业务时候，我们领取的排队号）
>
> IE浏览器中也有返回值，也是一个数字，但是数字有点跳跃（谷歌下刷新页面，定时器计数的起始值是一，IE下是从上次的结束值开始的，但是把页面关掉重新打开，所有浏览器都是从一开始的）

`clearInterval([number]) / clearTimeout([number])` 清除定时器的时候，我们只需要把一个数字传递进来，那么就清除了当前序号代表的这个定时器（不管你是用什么方法设置的，这两种方法都可以把定时器清除掉）

```
var timer1=setInterval(function(){
    clearTimeout(timer1);//->也可以把定时器清除掉，只不过为了保证语义化，我们最好使用哪个方法设置的，就使用哪个方法把它清除即可，这里最好使用clearInterval
},1000);
```

当我们把页面中的某一个定时器清除掉之后，后面排队号是不会发生改变的，当再设置一个新的定时器，会继续基于最后一个序号累加（其实就是我们去银行办业务的那点逻辑）

定时器中的递归思想

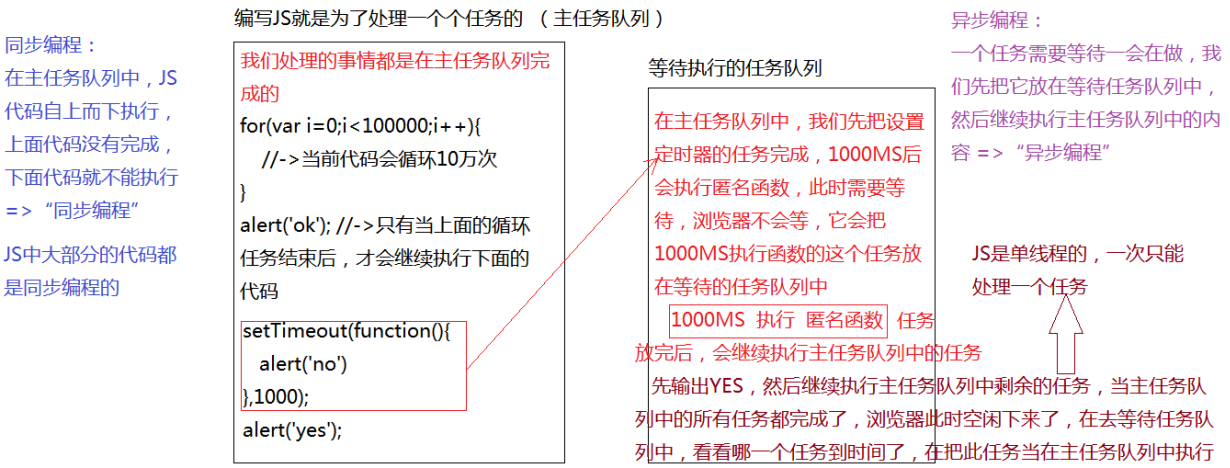
> 需求：每间隔一秒钟我们都在原来的基础上累加一，当累加到五的时候，结束这样的操作

```
//=>setInterval
var n = 0;
var timer = window.setInterval(function () {
    console.log(++n);
    if (n >= 5) {
        window.clearInterval(timer);
    }
}, 1000);

//=>setTimeout
//->定时器执行一次后就不再执行了
var n = 0;
var timer = window.setTimeout(function () {
    console.log(++n);
    if (n >= 5) {
        window.clearInterval(timer);
    }
}, 1000);

//->基于递归思想(函数执行的时候在调用自己执行)来实现我们的需求
var timer = null,
    n = 0;
function fn() {
    //->执行FN之前的第一步：把上一次没用的那个定时器清除掉
    window.clearTimeout(timer);

    console.log(++n);
    if (n >= 5) {
        //->已经到达最后的阶段了,结束后讲不再执行FN,我们也就没有必要在设置新的定时器了
        return;
    }
    timer = setTimeout(fn, 1000);//->每一次执行FN结束后,为了过一秒后再次执行FN,我们在重新设置一个定时器
}
timer = setTimeout(fn, 1000);
```

定时器的异步编程

> 所有的定时器都是异步编程

同步和异步编程




```
1. //=>定时器是异步编程
2. var n = 0;
3. window.setTimeout(function () {
4.     console.log(++n);//->2) 1
5. }, 1000);
6. console.log(n);//->1) 0
```

```
1.  //=>定时器的等待时间设置为零也不是立即执行:浏览器在处理一件事情的时候,会有一个最小的反应时间,我们写零浏览器是反应不过来的(谷歌一般最小的反应时间是5~6MS、IE是10~13MS,这个值会根据当前电脑CPU性能来决定的,每个人的不太一样)
2.  var n = 0;
3.  window.setTimeout(function () {
4.      console.log(++n);//->2) 1
5.  }, 0);//->5~6
6.  console.log(n);//->1) 0
7.
8.  window.onscroll = function () {
9.      //->通过ONSCROLL也可以鉴证浏览器是有自己的处理最小反应时间的,我们快速滑动,在短时间内触发的此处就少一些,慢一些滑动,滑动相同的距离用的时间就会多一些
10.     console.log('OK');
11. };
```

```
1.  //=>定时器设定了等待时间,到达时间后也不一定执行,看当前主任务队列中是否有任务正在执行呢,如果有任务在执行,到时间后也依然需要继续等待(JS是单线程的)
2.  var n = 0;
3.  window.setTimeout(function () {
4.      console.log(++n);//->2) 1
5.  }, 10);
6.
7.  // var strTime = new Date();
8.  for (var i = 0; i < 1000000000; i++) {
9.      //->循环10亿次
10. }
11. // var endTime = new Date();
12. // console.log(endTime - strTime);//->大约需要3057MS(通过这种方式可以监测代码执行的性能)
13.
14. console.log(n);//->1) 0
```

```
//=>此时的主任队列中遇到了死循环,浏览器永远空闲不下来,定时器等不到执行的那一天了(真实项目中要杜绝死循环:出现死循环就什么都做不了)
var n = 0;
window.setTimeout(function () {
    console.log(++n);
}, 10);
while (1 == 1) {
    //->死循环
}
```

```
//=>当主任队列任务完成完成后,会到等待任务队列中,把到时间的任务执行;如果很多等待的任务都到时间了,谁先到的,我们先执行谁;如果时间都很短,而且很相近,有些时候浏览器执行顺序混乱;
window.setTimeout(function () {
    console.log(1);//->3)
}, 100);

window.setTimeout(function () {
    console.log(2);//->1)
}, 0);

window.setTimeout(function () {
    console.log(3);//->2)
}, 50);

for (var i = 0; i < 1000000000; i++) {
    //->循环10亿次:需要大概3~4S,上面所有的定时器都到时间了
}
```

> 所有的事件绑定也都是异步编程

```
oImg.onload=function(){
    //->当图片加载成功后执行这个匿名函数
    alert('img is load success~');
}
oImg.onerror=function(){
    //->当图片加载失败后执行这个匿名函数
    alert('img is load error~');
}
alert('img is loading~');
```

> AJAX中也可以设置异步编程

```
var xhr=new XMLHttpRequest;
xhr.open('GET','地址',false); //->FALSE：同步    不写或者写TRUE：异步
...
```

JS中的同步(SYNC)或者异步(ASYNC)编程是一个非常重要的知识点，也是高级前端开发工程师应该必备的技能：`promise` 这个设计模式就是处理异步编程的

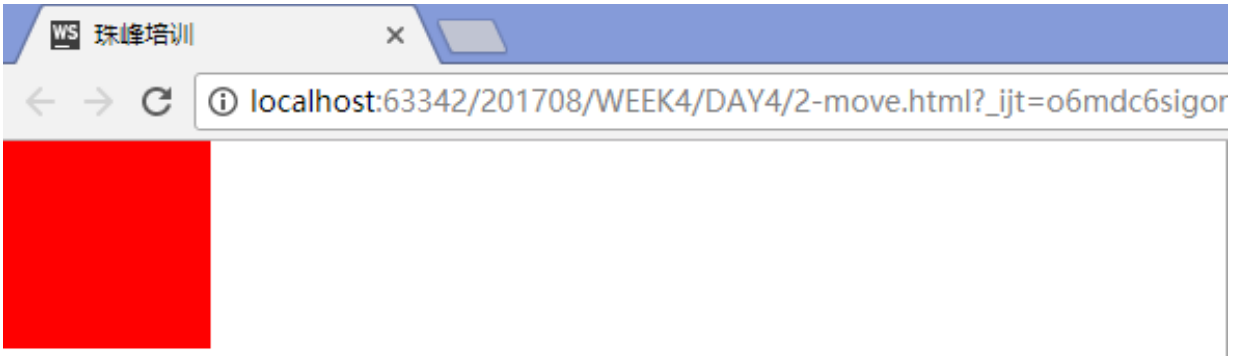
应用定时器实现JS动画

> 真正的项目中，我们的动画基本上就两种
>
> 1、不限定运动时间：规定的是步长（每隔多长时间走多远），我们控制需要运动的元素在现有位置的基础上累加步长即可
>
> 2、限定运动时间：在规定时间内完成我们的动画，这个就需要我们获取总距离、总时间等信息，然后按照运动公式来处理了

固定步长的匀速运动

让红色的小盒子运动到右边界的位置:修改当前元素的LEFT样式值

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>珠峰培训</title>
    <link rel="stylesheet" href="css/reset.min.css">
    <style>
        .box {
            position: absolute;
            left: 0; /*不加LEFT,IE获取的LEFT是AUTO,谷歌下获取的是0*/
            width: 100px;
            height: 100px;
            background: red;
        }
    </style>
</head>
<body>
<div class="box" id="box"></div>

<script src="js/utils201708.js"></script>
<script src="js/2-2.js"></script>
</body>
</html>
```




```
var minL = 0,
    maxL = utils.win('clientWidth') - box.offsetWidth;

//->真实项目中为了避免全局变量的污染:我们定时器的返回值不要定义为全局的变量,而是设定在当前元素的自定义属性上(而且在任何的作用域中或者任何的位置,如果有需要我们都可以通过自定义属性的方式获取到这个结果)
box.timer = window.setInterval(function () {
    var curL = utils.css(box, 'left');
     //->为了防止多走一步会超过边界,少走一步到不了边界,我们JS实现动画的时候,边界判断都是加上步长来处理的(相当于模拟走一步看情况,如果模拟这一步超过了边界,我们直接让其运动到边界即可)
    if (curL + 10 >= maxL) {
        utils.css(box, 'left', maxL);
        window.clearInterval(box.timer);
        return;
    }
    curL += 10;
    utils.css(box, 'left', curL);
}, 17);
```

```
//=>使用setTimeout实现这个需求
var minL = 0,
    maxL = utils.win('clientWidth') - box.offsetWidth;
function move() {
    //->每一次执行方法之前都把上一次没用的定时器清除掉(优化内存)
    window.clearTimeout(box.timer);

    var curL = utils.css(box, 'left');
    if (curL + 10 >= maxL) {
        utils.css(box, 'left', maxL);
        return;
    }
    curL += 10;
    utils.css(box, 'left', curL);

    //->方法执行完成后都会重新的设置一个定时器,让其17MS后重新执行这个方法(递归思想)
    box.timer = window.setTimeout(move, 17);
}
move();
```

基于第一个需求：实现红色盒子在两个边界之间来回的反弹

```
var minL = 0,
    maxL = utils.win('clientWidth') - box.offsetWidth;

move(maxL);

function move(target) {
    //->首先区分是向左还是向右
    //1、如果当前的LEFT值<=目标值TARGET（向右）
    //2、如果当前的LEFT值>=目标值TARGET（向左）
    var curL = utils.css(box, 'left'),
        dir = curL <= target ? 'right' : 'left';

    //->设置定时器实现运动即可
    var step = 10;
    box.timer = window.setInterval(function () {
        var curL = utils.css(box, 'left'),
            isEnd = false;
        dir === 'right' ? (curL + step >= target ? isEnd = true : null) : (curL - step <= target ? isEnd = true : null);
        if (isEnd) {
            utils.css(box, 'left', target);
            window.clearInterval(box.timer);
            //->控制下一次运动
            dir === 'right' ? move(minL) : move(maxL);
            return;
        }
        utils.css(box, 'left', dir === 'right' ? curL + step : curL - step);
    }, 17);
}
```

限定时间的匀速动画

需求还是沿袭上面的需求，只不过规定从左边界到达右边界总时间需要1000MS

//=>总时间1000MS D
//=>起始位置 B
//=>总距离：目标位置-起始位置 C
//=>已经走的时间 T

限定时间的匀速动画：就是随时获取到当前元素的位置即可，让元素运动到这个位置，一直到总时间结束，就完成了动画

- T/D：已经走过的时间占总时间的百分比(我们已经走过百分之多少了)
- T/D*C：已经走过的百分比乘以总距离=已经走过的具体距离(我们已经走了多远)
- T/D*C+B：已经走过的距离+起始的位置=当前的位置(当前我们应该在哪)

```
//->匀速动画公式：获取当前元素应有的位置
//->t:time 已经走过的时间
//->b:begin 当前元素起始位置
//->c:change 要运动的总距离
//->d:duration 动画的总时间
function Linear(t,b,c,d){
    return t / d * c + b;
}
```

```
var time = 0,
    begin = utils.css(box, 'left'),
    target = utils.win('clientWidth') - box.offsetWidth,
    change = target - begin,
    duration = 5000;
box.timer = window.setInterval(function () {
    time += 17;
    //->当到达总时间的时候,结束动画
    if (time >= duration) {
        utils.css(box, 'left', target);
        window.clearInterval(box.timer);
        return;
    }
    //->获取当前元素的位置,并且让元素运动到这个位置
    var curL = Linear(time, begin, change, duration);
    utils.css(box, 'left', curL);
}, 17);
```

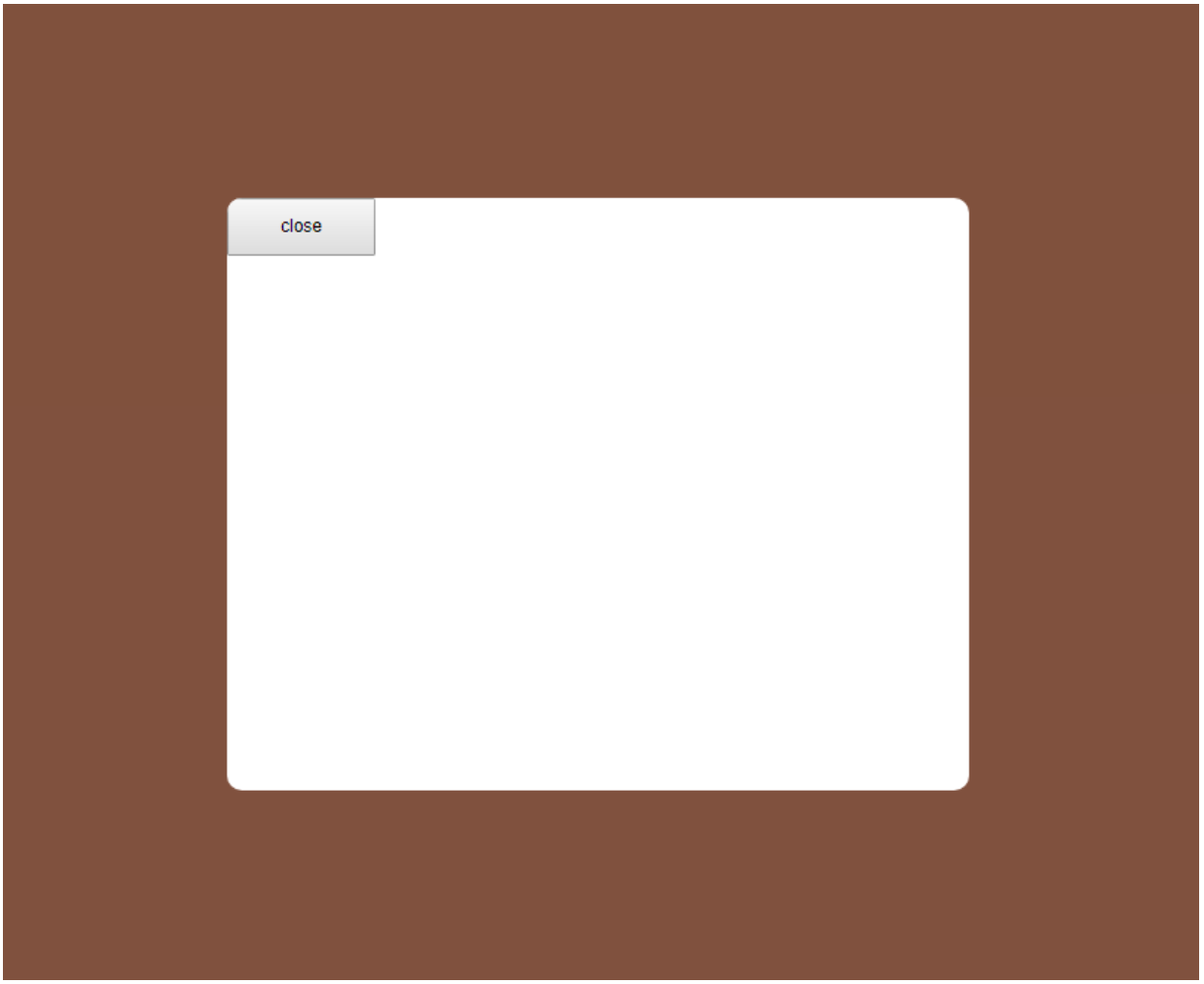
多方向匀速动画

> 基于上面的需求，实现从左上角到右下角或者实现当前元素多方向匀速运动

```
//=>封装一个动画库:实现当前元素限定时间内的多方向匀速运动
~function () {
    //=>匀速运动的动画公式
    function Linear(t, b, c, d) {
        return t / d * c + b;
    }

    //=>封装一个实现动画的方法
    //->curEle：当前要实现运动的元素
    //->target：要运动的目标位置 {xxx:xxx,xxx:xxx...}
    //->duration：运动的总时间,不传递默认是1000MS
    function animate(curEle, target, duration) {
        //1、获取T/B/C/D
        duration = duration || 1000;
        var time = 0,
            begin = {},
            change = {};
        for (var key in target) {
            if (target.hasOwnProperty(key)) {
                begin[key] = utils.css(curEle, ke
y);
                change[key] = target[key] - begi
n[key];
            }
        }

        //2、实现动画
        clearInterval(curEle.animateTimer);//->在设
置新动画之前,把正在运行的其它动画都清除掉,防止多动画之间的冲
突
        curEle.animateTimer = setInterval(function
() {
            time += 17;
            //->结束动画
            if (time >= duration) {
                utils.css(curEle, target);
                clearInterval(curEle.animateTime
r);
```

```
                return;
            }

            //->通过匀速公式获取每个方向的当前位置,让元素运动到这个位置
            var current = {};
            for (var key in target) {
                if (target.hasOwnProperty(key)) {
                    current[key] = Linear(time, begin[key], change[key], duration);
                }
            }
            utils.css(curEle, current);
        }, 17);
    }

    //=>暴露到全局使用
    window.zhufengAnimate = animate;
}();

zhufengAnimate(box, {
    top: 300,
    left: 500,
    width: 10,
    height: 10,
    opacity: 0.2
}, 5000);

zhufengAnimate(box, {
    width: 300,
    height: 300,
    opacity: 0.2
}, 500);
```

案例：弹出框



HTML && CSS

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>珠峰培训</title>
    <!--import css-->
    <link rel="stylesheet" href="css/reset.min.css">
    <link rel="stylesheet" href="css/dialog.css">
</head>
<body>
<!--dialog-->
<div class="dialogBg" id="dialogBg"></div>
<div class="dialog" id="dialog">
    <input type="button" value="close" id="dialogClose">
</div>

<!--import js-->
<script src="js/utils201708.min.js"></script>
<script src="js/animate.js"></script>
<script src="js/dialog.js"></script>
</body>
</html>
```

`dialog.css`

```
1. html, body {
2.     width: 100%;
3.     height: 1000%;
4.     background: -webkit-linear-gradient(top left,
   lightsalmon, lightpink, lightcoral, lightblue);
5. }
6.
7. .dialogBg {
8.     display: none;
9.     position: fixed;
10.    top: 0;
11.    left: 0;
12.    z-index: 1000;
13.    width: 100%;
14.    height: 100%;
15.    background: #000;
16.    opacity: 0.5;
17.    filter: alpha(opacity=50);
18. }
19.
20. .dialog {
21.    display: none;
22.    position: fixed;
23.    top: 50%;
24.    left: 50%;
25.    z-index: 2000;
26.    /*margin: -200px 0 0 -250px;
27.    width: 500px;
28.    height: 400px;*/
29.    margin: 0;
30.    width: 0;
31.    height: 0;
32.    opacity: 0;
33.    filter: alpha(opacity=0);
34.
35.    background: #FFF;
36.    border-radius: 10px;
37.    overflow: hidden;
```

```
}

.dialog input[type='button'] {
    display: block;
    width: 100px;
    line-height: 35px;
}
```

`dialog.js`

```
var dialogBg = document.getElementById('dialogBg'),
    dialog = document.getElementById('dialog'),
    dialogClose = document.getElementById('dialogClose');

document.body.onclick = function () {
    //->让弹出层显示
    utils.css(dialogBg, 'display', 'block');
    utils.css(dialog, 'display', 'block');

    //->让显示内容的提示框区域是从中间放大出来的
    zhufengAnimate(dialog, {
        width: 500,
        height: 400,
        marginLeft: -250,
        marginTop: -200,
        opacity: 1
    }, 200);
};

dialogClose.onclick = function (e) {
    utils.css(dialogBg, 'display', 'none');

    //->首先让弹出层先慢慢的消失
    zhufengAnimate(dialog, {
        width: 0,
        height: 0,
        marginLeft: 0,
        marginTop: 0,
        opacity: 0
    }, 200);
    //->问题：当我们动画结束后，应该让其display=none才可以

    e = e || window.event;
    e.stopPropagation ? e.stopPropagation() : e.cancelBubble = true;//->阻止事件的传播
```

```
35.  };
```

非匀速运动

> 在匀速动画基础上支持非匀速的运动公式即可，继续完善我们的animate动画库

```
~function () {
    //=>珠峰培训TWEEN算法动画公式(除了匀速还有非匀速的公式)
    //=>http://old.zhufengpeixun.cn/tween/
    var animationEffect = {
        Linear: function (t, b, c, d) {
            return c * t / d + b;
        },
        Bounce: {
            easeIn: function (t, b, c, d) {
                return c - animationEffect.Bounce.easeOut(d - t, 0, c, d) + b;
            },
            easeOut: function (t, b, c, d) {
                if ((t /= d) < (1 / 2.75)) {
                    return c * (7.5625 * t * t) + b;
                } else if (t < (2 / 2.75)) {
                    return c * (7.5625 * (t -= (1.5 / 2.75)) * t + .75) + b;
                } else if (t < (2.5 / 2.75)) {
                    return c * (7.5625 * (t -= (2.25 / 2.75)) * t + .9375) + b;
                } else {
                    return c * (7.5625 * (t -= (2.625 / 2.75)) * t + .984375) + b;
                }
            },
            easeInOut: function (t, b, c, d) {
                if (t < d / 2) {
                    return animationEffect.Bounce.easeIn(t * 2, 0, c, d) * .5 + b;
                }
                return animationEffect.Bounce.easeOut(t * 2 - d, 0, c, d) * .5 + c * .5 + b;
            }
        },
        Quad: {
```

```
            easeIn: function (t, b, c, d) {
                return c * (t /= d) * t + b;
            },
            easeOut: function (t, b, c, d) {
                return -c * (t /= d) * (t - 2) +
b;
            },
            easeInOut: function (t, b, c, d) {
                if ((t /= d / 2) < 1) {
                    return c / 2 * t * t + b;
                }
                return -c / 2 * ((--t) * (t - 2) -
1) + b;
            }
        },
        Cubic: {
            easeIn: function (t, b, c, d) {
                return c * (t /= d) * t * t + b;
            },
            easeOut: function (t, b, c, d) {
                return c * ((t = t / d - 1) * t *
t + 1) + b;
            },
            easeInOut: function (t, b, c, d) {
                if ((t /= d / 2) < 1) {
                    return c / 2 * t * t * t + b;
                }
                return c / 2 * ((t -= 2) * t * t +
2) + b;
            }
        },
        Quart: {
            easeIn: function (t, b, c, d) {
                return c * (t /= d) * t * t * t +
b;
            },
            easeOut: function (t, b, c, d) {
                return -c * ((t = t / d - 1) * t *
t * t - 1) + b;
```

```
            },
            easeInOut: function (t, b, c, d) {
                if ((t /= d / 2) < 1) {
                    return c / 2 * t * t * t * t + b;
                }
                return -c / 2 * ((t -= 2) * t * t * t - 2) + b;
            }
        },
        Quint: {
            easeIn: function (t, b, c, d) {
                return c * (t /= d) * t * t * t * t + b;
            },
            easeOut: function (t, b, c, d) {
                return c * ((t = t / d - 1) * t * t * t * t + 1) + b;
            },
            easeInOut: function (t, b, c, d) {
                if ((t /= d / 2) < 1) {
                    return c / 2 * t * t * t * t * t + b;
                }
                return c / 2 * ((t -= 2) * t * t * t * t + 2) + b;
            }
        },
        Sine: {
            easeIn: function (t, b, c, d) {
                return -c * Math.cos(t / d * (Math.PI / 2)) + c + b;
            },
            easeOut: function (t, b, c, d) {
                return c * Math.sin(t / d * (Math.PI / 2)) + b;
            },
            easeInOut: function (t, b, c, d) {
                return -c / 2 * (Math.cos(Math.PI
```

```
    * t / d) - 1) + b;
95.             }
96.         },
97.         Expo: {
98.             easeIn: function (t, b, c, d) {
99.                 return (t == 0) ? b : c * Math.po
    w(2, 10 * (t / d - 1)) + b;
100.             },
101.             easeOut: function (t, b, c, d) {
102.                 return (t == d) ? b + c : c * (-Ma
    th.pow(2, -10 * t / d) + 1) + b;
103.             },
104.             easeInOut: function (t, b, c, d) {
105.                 if (t == 0) return b;
106.                 if (t == d) return b + c;
107.                 if ((t /= d / 2) < 1) return c / 2
    * Math.pow(2, 10 * (t - 1)) + b;
108.                 return c / 2 * (-Math.pow(2, -10 *
    --t) + 2) + b;
109.             }
110.         },
111.         Circ: {
112.             easeIn: function (t, b, c, d) {
113.                 return -c * (Math.sqrt(1 - (t /=
    d) * t) - 1) + b;
114.             },
115.             easeOut: function (t, b, c, d) {
116.                 return c * Math.sqrt(1 - (t = t /
    d - 1) * t) + b;
117.             },
118.             easeInOut: function (t, b, c, d) {
119.                 if ((t /= d / 2) < 1) {
120.                     return -c / 2 * (Math.sqrt(1 -
    t * t) - 1) + b;
121.                 }
122.                 return c / 2 * (Math.sqrt(1 - (t -
    = 2) * t) + 1) + b;
123.             }
124.         },
```

```
125.         Back: {
126.             easeIn: function (t, b, c, d, s) {
127.                 if (s == undefined) s = 1.70158;
128.                 return c * (t /= d) * t * ((s + 1)
    * t - s) + b;
129.             },
130.             easeOut: function (t, b, c, d, s) {
131.                 if (s == undefined) s = 1.70158;
132.                 return c * ((t = t / d - 1) * t *
    ((s + 1) * t + s) + 1) + b;
133.             },
134.             easeInOut: function (t, b, c, d, s) {
135.                 if (s == undefined) s = 1.70158;
136.                 if ((t /= d / 2) < 1) {
137.                     return c / 2 * (t * t * (((s
    *= (1.525)) + 1) * t - s)) + b;
138.                 }
139.                 return c / 2 * ((t -= 2) * t *
    (((s *= (1.525)) + 1) * t + s) + 2) + b;
140.             }
141.         },
142.         Elastic: {
143.             easeIn: function (t, b, c, d, a, p) {
144.                 if (t == 0) return b;
145.                 if ((t /= d) == 1) return b + c;
146.                 if (!p) p = d * .3;
147.                 var s;
148.                 !a || a < Math.abs(c) ? (a = c, s
    = p / 4) : s = p / (2 * Math.PI) * Math.asin(c /
    a);
149.                 return -(a * Math.pow(2, 10 * (t -
    = 1)) * Math.sin((t * d - s) * (2 * Math.PI) / p))
    + b;
150.             },
151.             easeOut: function (t, b, c, d, a, p) {
152.                 if (t == 0) return b;
153.                 if ((t /= d) == 1) return b + c;
154.                 if (!p) p = d * .3;
155.                 var s;
```

```
156.                 !a || a < Math.abs(c) ? (a = c, s
    = p / 4) : s = p / (2 * Math.PI) * Math.asin(c /
    a);
157.                 return (a * Math.pow(2, -10 * t) *
    Math.sin((t * d - s) * (2 * Math.PI) / p) + c +
    b);
158.             },
159.             easeInOut: function (t, b, c, d, a, p)
    {
160.                 if (t == 0) return b;
161.                 if ((t /= d / 2) == 2) return b +
    c;
162.                 if (!p) p = d * (.3 * 1.5);
163.                 var s;
164.                 !a || a < Math.abs(c) ? (a = c, s
    = p / 4) : s = p / (2 * Math.PI) * Math.asin(c /
    a);
165.                 if (t < 1) return -.5 * (a * Mat
    h.pow(2, 10 * (t -= 1)) * Math.sin((t * d - s) *
    (2 * Math.PI) / p)) + b;
166.                 return a * Math.pow(2, -10 * (t -=
    1)) * Math.sin((t * d - s) * (2 * Math.PI) / p) *
    .5 + c + b;
167.             }
168.         }
169.     };
170.
171.     //=>realization animate (depend on utils)
172.     function animate(options) {
173.         //->initialize(init) parameters
174.         var _default = {
175.             curEle: null,
176.             target: null,
177.             duration: 1000,
178.             effect: animationEffect.Linear
179.         };
180.         for (var attr in options) {
181.             if (options.hasOwnProperty(attr)) {
182.                 _default[attr] = options[attr];
```

```
            }
        }
        var curEle = _default.curEle,
            target = _default.target,
            duration = _default.duration,
            effect = _default.effect;

        //->prepare T/B/C/D
        var time = 0,
            begin = {},
            change = {};
        for (var key in target) {
            if (target.hasOwnProperty(key)) {
                begin[key] = utils.css(curEle, ke
y);
                change[key] = target[key] - begi
n[key];
            }
        }

        //->running (clear other animate before ru
nning)
        clearInterval(curEle.animateTimer);
        curEle.animateTimer = setInterval(function
() {
            time += 17;
            if (time >= duration) {
                utils.css(curEle, target);
                clearInterval(curEle.animateTime
r);
                return;
            }
            var current = {};
            for (var key in target) {
                if (target.hasOwnProperty(key)) {
                    current[key] = effect(time, be
gin[key], change[key], duration);
                }
            }

```

```
            utils.css(curEle, current);
        }, 17);
    }

    //=>setting window property
    window.zhufengEffect = animationEffect;
    window.zhufengAnimate = animate;
}();
```

回调函数CALLBACK

> 回调函数：把一个函数(B)当做‘实参’，传递给另外一个执行的函数(A)，在A执行的过程中，根据需求把B执行

```
function fn(callBack){
    //->callBack:我们传递进来的这个匿名函数
    //callBack();
    //->为了防止callBack不传递函数值,执行会报错,我们在回调函数执行的时候一般都会加判断,只有传递的是函数在执行
    //typeof callBack==='function'?callBack():null;
    callBack && callBack();
}
fn(function(){
    //->把匿名函数当做一个实参传递给FN
    console.log('ok');
    //->this:window 回调函数中的THIS一般都是window(严格模式:undefined)
});

fn();//->callBack:undefined
```

> 回调函数的小应用：完善动画库，让动画库支持动画完成后执行相关的事宜

```
function animate(options) {
    var _default = {
        ...
        callBack: null
    };
    ...
    var callBack = _default.callBack;
    ...
    curEle.animateTimer = setInterval(function ()
{
        if (time >= duration) {
           ...
           //->run end: perform callback functions
           callBack && callBack.call(curEle);
            return;
        }
        ...
    },17);
```

```
//=>当动画完成的时候：让当前元素的背景颜色变为PINK，让其透明度变为0.5，....
zhufengAnimate({
    curEle: box,
    target: {
        top: 300,
        left: 1000,
        width: 150,
        height: 150
    },
    effect: zhufengEffect.Bounce.easeOut,
    duration: 500,
    callBack: function () {
        //->把我们需要在动画完成处理的事情都放在这个函数中,在动画库中,动画完成后只需要把传递的这个回调函数执行即可
        //->this:box
        utils.css(this, {
            background: 'green',
            opacity: 0.5
        });
    }
});
```

> 回调函数B可以在宿主函数A(B在A中执行的,所以可以把A叫做B的宿主函数)的`任何位置执行`(根据需求而定)，而且还可以`执行零到N次`，还可以给回调函数B`传递参数`或者`改变里面的THIS`，也可以`接收回调函数B的返回值`，进行后续的相关处理
>
> 回调函数可以在宿主环境中‘肆无忌惮’的像正常的普通函数一样执行

`sort / forEach / map / filter ...`

```
1. //->数组中的很多方法都是支持回调函数的，例如：sort中传递的匿名函数就是回调函数
2. ary.sort(function(a,b){
3.     //->SORT执行中,把传递的回调函数执行,不仅执行还给回调传递了两个实参的值
4.     return a-b;//->同样也接收了回调函数的返回结果,如果回调返回的是>0的值,SORT方法中会让当前项和后一项交换位置(SORT的原理)
5. });
6. 
7. ary.forEach(function(item,index,input){
8.     //->数组中有多少项，就会把当前的回调函数执行多少次，不仅执行，还会给它传递参数：
9.     //item：当前遍历的这一项
10.    //index：当前遍历项的索引
11.    //input：遍历的原始数组
12. 
13.    //->但是这个方法不支持回调函数的返回值(写了RETURN也没用)
14. 
15. },[context]);//->第二个传递给FOREACH的参数是改变回调函数中的THIS指向
16. 
17. 
18. ary.map(function(item,index,input){
19.     //->MAP和FOREACH基本一样
20.     //->区别是MAP传递的回调函数支持返回值,我们RETURN是啥,相当于把当前的遍历项替换成啥
21.     return item*10; //->让原有的每一项乘以10
22. },[context]);
```

`replace`

```
1. //->当前正则和字符串匹配几次，我们的回调函数执行几次，并且把本次捕获的结果，传递给回调函数，回调函数中的返回值会把当前正则匹配的这一项替换掉
2. str.replace([reg],[function]);
```

`setInterval([function]...) / setTimeout([function]...)`

重写forEach方法，实现兼容

```
1. Array.prototype.myForEach = function myForEach() {
2.     //->this:需要处理的数组
3.     var callBack = arguments[0],
4.         context = arguments[1];
5.     //->兼容
6.     if ('forEach' in Array.prototype) {
7.         this.forEach(callBack, context);
8.         return;
9.     }
10.
11.     //->不兼容(IE6~8)
12.     for (var i = 0; i < this.length; i++) {
13.         callBack && callBack.call(context, this[i], i, this);
14.     }
15. };
16.
17. var obj = {name: 'zhufeng'};
18. [12, 23, 34, 45].myForEach(function (item, index, input) {
19.     console.log(item, index, input);
20. }, obj);
```

> 思考题：重写数组的map方法，也可以尝试重写字符串的replace方法