[笔记][LIKE-JS][7-JS进阶-缓动动画]

JavaScript

[笔记][LIKE-JS][7-JS进阶-缓动动画]
- 180. Math常用的函数(了解)
- 181. 基本缓动动画(掌握)
- 182. 封装缓动动画函数(掌握)
- 184. JS获取CSS的属性值(掌握)
- 185. 封装缓动动画函数-单值(掌握)
- 186. JSON的遍历(掌握)
- 187. 封装缓动动画函数-多值(掌握)
- 188. 封装缓动动画函数-回调(掌握)
- 189. 封装缓动动画函数-透明度(掌握)
- 190. 缓动案例-联动特效(掌握)
- 191. 缓动案例-楼层特效-上(掌握)
- 192. 缓动案例-楼层特效-中(掌握)
- 193. 缓动案例-楼层特效-下(掌握)
- 194. 缓动案例-楼层特效-结尾(掌握)
- 195. 缓动案例-音乐导航-上(掌握)
- 196. 缓动案例-音乐导航-下(掌握)
- 197. 网易轮播图-上(掌握)
- 198. 网易轮播图-样式-1(掌握)
- 199. 网易轮播图-动态效果-1(掌握)
- 200. 网易轮播图-动态效果-1(掌握)
- 201. 网易轮播图-动态效果-2(掌握)
- 202. 网易轮播图-动态效果-3(掌握)
- 203. 网易轮播图-动态效果-4(掌握)
- 204. 旋转木马特效-前奏(掌握)
- 205. 旋转木马特效-界面布局(掌握)
- 206. 旋转木马特效-动态特效-上(掌握)
- 207. 旋转木马特效-动态特效-下(掌握)
- 208. 面向对象-知识点回顾
- 209. 面向对象-构造函数-回顾
- 210. 面向对象-矩形

180. Math常用的函数(了解)

```
<script>
    console.log(parseInt("20.20"));  // 20
    console.log(parseFloat("21.88嘿嘿嘿"));  // 21.88
    console.log(typeof parseFloat("21.88嘿嘿嘿"));  // number

    // Math
    // 1. 四舍五入
    console.log(Math.round(10.61));  // 11

    // 2. 天花板
    console.log(Math.ceil(10.01));  // 11

    // 3. 地板
    console.log(Math.floor(8.888));  // 8

    // 4. 取绝对值
    console.log(Math.abs(-12.11));  // 12.11

</script>
```

---

181. 基本缓动动画(掌握)

缓动动画原理：盒子本身的位置 + 步长（不断变化的步长）
公式：`begin = begin + (end - begin) * 缓动系数`

---

182. 封装缓动动画函数(掌握)

2.1 常见的 js访问 CSS 属性

- 在开发中，访问得到css 属性，比较常用的有两种：
- ◦ 点语法

  box.style.width，box.style.height，box.style.top，box.style.left 得到带有单位的属性值，比如：200px； 但是，点语法存在一个很致命的问题，跟在style后面的属性不能由外面传入

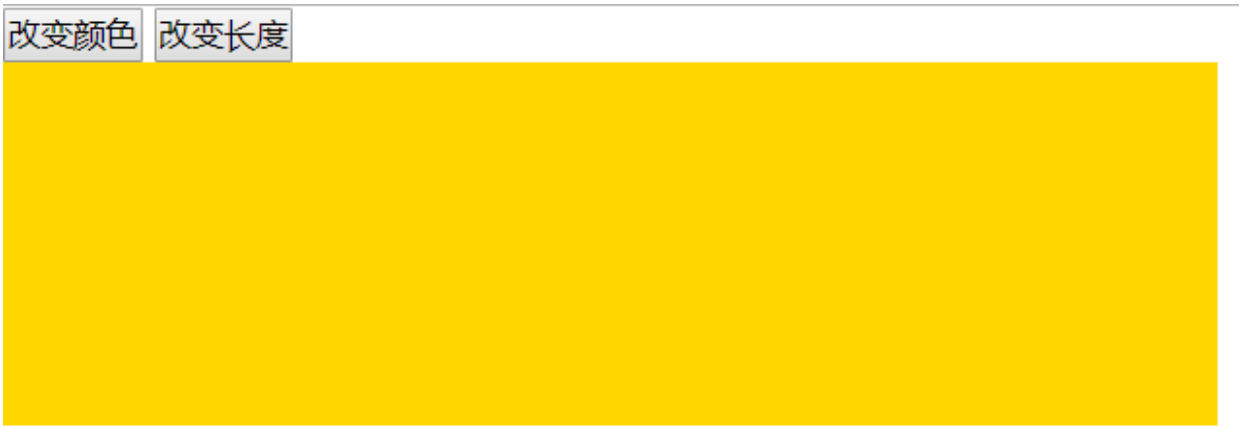
```
var h = 'height';
box.style.h = 300 + 'px';
```

- ◦ 下标语法

  利用 [] 访问属性 元素.style[“属性”];

```
这种语法的好处就是可以动态的传递参数作为属性
```

```
var h = 'height';
box.style[h] = 300 + 'px';
```




```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <style>
        *{
            margin: 0;
            padding: 0;
        }

        #box{
            width: 150px;
            height: 150px;
            background-color: skyblue;
        }
    </style>
</head>
```

```
<body>
    <button id="btn1">改变颜色</button>
    <button id="btn2">改变长度</button>
    <div id="box"></div>
    <script>
        var btn1 = document.getElementById("btn1");
        var btn2 = document.getElementById("btn2");
        var box = document.getElementById("box");

        btn1.onclick = function () {
            changeCssStyle(box, "backgroundColor", "gold");
        };

        btn2.onclick = function () {
            changeCssStyle(box, "width", "500px");
        };

        function changeCssStyle(obj, attr, value) {
            obj.style[attr] = value;
        }
    </script>
</body>
</html>
```

184. JS获取CSS的属性值(掌握)

2.2 JS获取CSS的样式

> 在开发中，我们想要获得css 的样式，通常采用：box.style.top ，box.style.backgorundColor等， 但这种方式只能得到 行内样式，而平常用的最多的是页内样式或者外部样式， 那我们如何解决这样的问题？

- 在IE和Opera浏览器

  obj.currentStyle

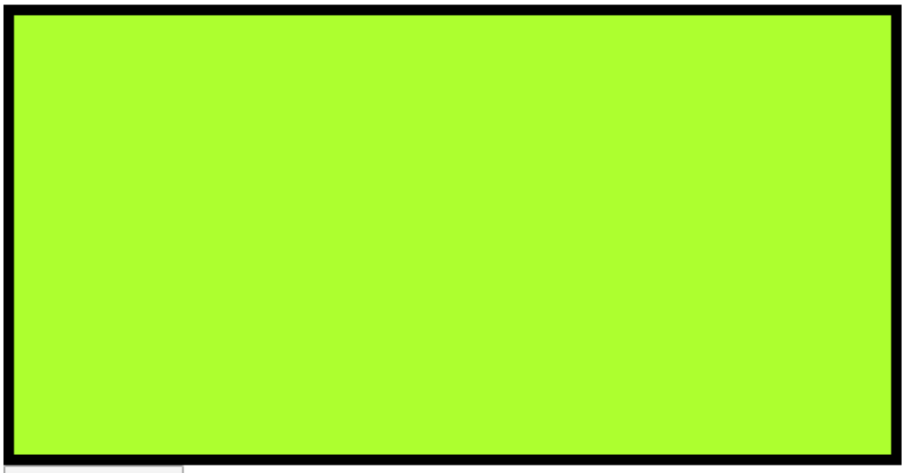
  - 其他W3C标准浏览器

    window.getComputedStyle("元素", "伪类")（ 注意：两个选项是必须的， 没有伪类 用 **null** 替代 ）

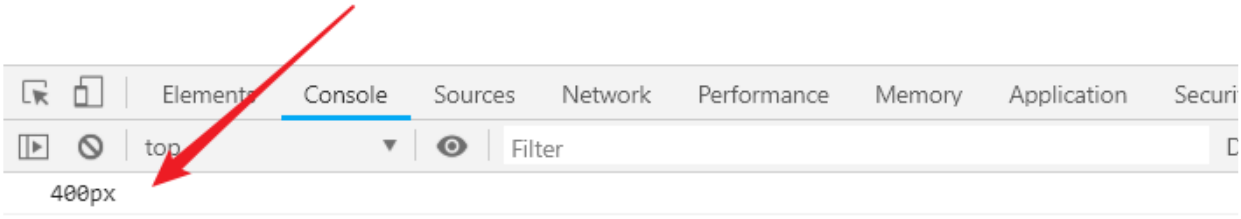
- 兼容写法

```
function getStyleAttr(obj, attr) {
    if(obj.currentStyle){ // IE 和 opera
         return obj.currentStyle[attr];
    }else {
         return window.getComputedStyle(obj, null)[attr];
    }
}
```

验证是否 `box.style.top` 这样是否真的拿不到页内和外部样式



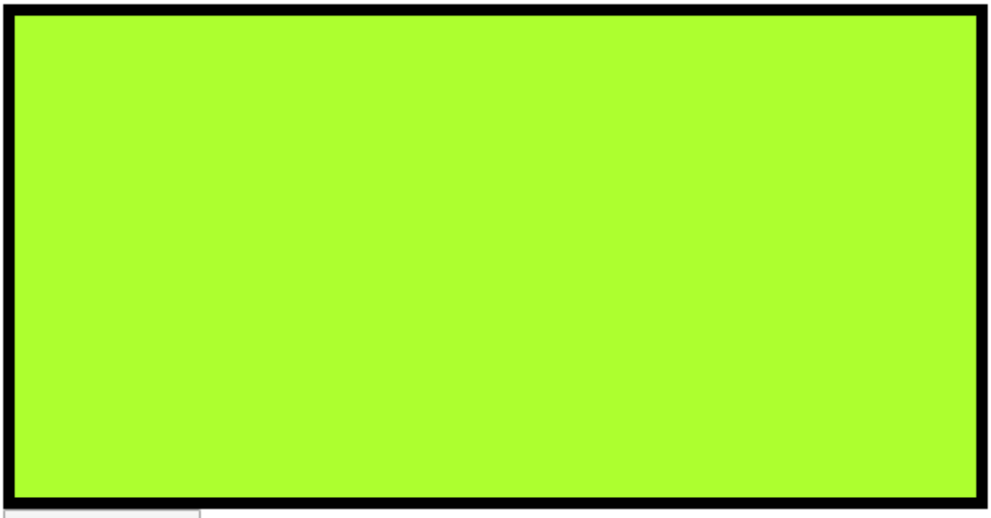
获取属性值



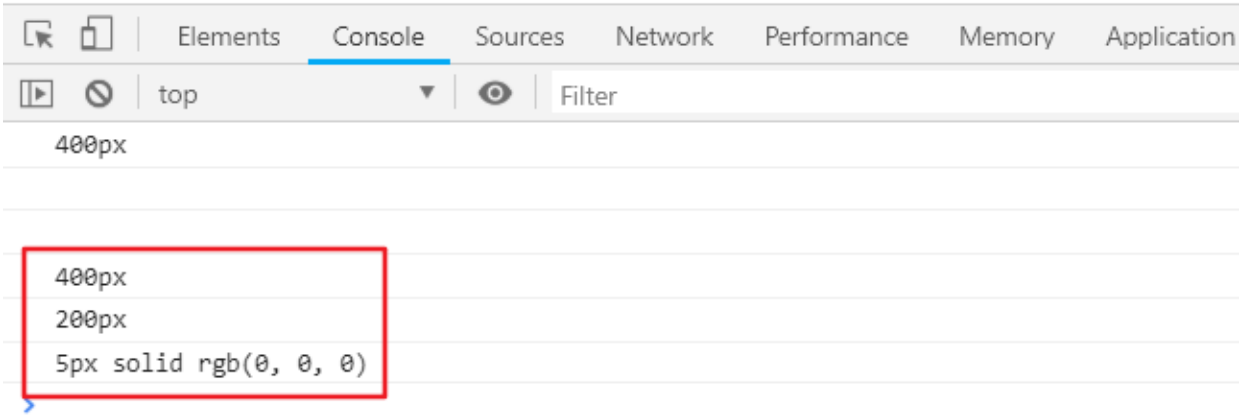
只拿到了行内样式！！！

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <style>
        #box{
            width: 200px;
            height: 200px;
            background-color: greenyellow;
            border: 5px solid #000;
        }
    </style>
</head>
```

```
<body>
    <div id="box" style="width: 400px;"></div>
    <button id="btn">获取属性值</button>
    <script>
        var box = document.getElementById("box");
        var btn = document.getElementById("btn");
        btn.onclick = function () {
            console.log(box.style.width);
            console.log(box.style.height);
            console.log(box.style.border);
        };
    </script>
</body>
</html>
```



获取属性值




```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <style>
        #box{
            width: 200px;
            height: 200px;
```

```
            background-color: greenyellow;
            border: 5px solid #000;
        }
    </style>
</head>
<body>
    <div id="box" style="width: 400px;"></div>
    <button id="btn">获取属性值</button>
    <script>
        var box = document.getElementById("box");
        var btn = document.getElementById("btn");
        btn.onclick = function () {
            console.log(box.style.width);
            console.log(box.style.height);
            console.log(box.style.border);

            console.log(getStyleAttr(box, "width"));
            console.log(getStyleAttr(box, "height"));
            console.log(getStyleAttr(box, "border"));
        };

        function getStyleAttr(obj, attr) {
            if (obj.currentStyle) {  // IE 和 Opera
                return obj.currentStyle[attr];
            } else {
                return window.getComputedStyle(obj, null)[attr];
            }
        }
    </script>
</body>
</html>
```

最后封装好的函数：

```
/**
 * 获取CSS样式值
 * @param {object}obj
 * @param {string}attr
 * @returns {string}
 */
function getCssStyleValue(obj, attr) {
    if (obj.currentStyle) {  // IE 和 Opera
        return obj.currentStyle[attr];
    } else {
        return window.getComputedStyle(obj, null)[attr];
    }
}
```

185. 封装缓动动画函数-单值(掌握)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <style>
        * {
            margin: 0;
            padding: 0;
        }

        #box {
            width: 100px;
            height: 100px;
            background-color: red;
            position: absolute;
        }
    </style>
</head>
<body>
<button id="btn">往右走</button>
<button id="btn1">往左走</button>
<div id="box"></div>
<script src="js/myfuncs.js"></script>
<script>
    window.onload = function () {
        // 0. 定义变量
        var box = $("box");

        // 1. 监听按钮的点击
        $("btn").onclick = function () {
            buffer(box, "left", 800);
        };
        $("btn1").onclick = function () {
            buffer(box, "height", 200);
        };
    };

    function buffer(obj, attr, target) {
        // 1.1 清除定时器
        clearInterval(obj.timer);

        // 1.2 设置定时器
        obj.timer = setInterval(function () {
            // 1.3.1 获取初始值
            var begin = parseInt(getCssStyleValue(obj, attr));
            // 1.3 求出步长
            var speed = (target - begin) * .2;
```

```
            // 判断是否向上取整
            speed = target > begin ? Math.ceil(speed) : Math.floor(spee
d);
            // 1.4 动起来
            obj.style[attr] = begin + speed + "px";
            obj.innerText = begin;
            // 1.5 判断
            if (begin === target) {
                clearInterval(obj.timer);
            }
        }, 20);
    }
</script>
</body>
</html>
```

186. JSON的遍历(掌握)

2.3 JSON 遍历

- for in 关键字

  for (变量 in 对象) { 执行语句; }

```
for(var k in json){
   console.log(k); // key
   console.log(json[k]); // value
}
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<script>
    var dataArr = [10, 20, 50, 100];
    dataArr.forEach(function (item, index, arr) {
        // console.log(item, index);
    });
```

```
        var obj = {left: 300, top: 200, width: 500};
        // 对象没有 forEach 方法
        for (var key in obj) {
            console.log(key);

            // 点语法拿不到!
            // console.log(obj.key);
            // 拿不到的原因是，key 是字符串的类型
            console.log(typeof key);  // string

            // 只能用下标语法取
            console.log(obj[key]);
        }

    </script>
    </body>
    </html>
```

187. 封装缓动动画函数-多值(掌握)

往右走 往左走

```
    <!DOCTYPE html>
    <html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>Title</title>
        <style>
```

```
        * {
            margin: 0;
            padding: 0;
        }

        #box {
            width: 100px;
            height: 100px;
            background-color: red;
            position: absolute;
        }
    </style>
</head>
<body>
<button id="btn">往右走</button>
<button id="btn1">往左走</button>
<div id="box"></div>
<script src="js/myfuncs.js"></script>
<script>
    window.onload = function () {
        // 0. 定义变量
        var box = $("box");

        // 1. 监听按钮的点击
        $("btn").onclick = function () {
            buffer(box, {"left": 800, "top": 120, "width": 230, "height":
 400});
        };
        $("btn1").onclick = function () {
            buffer(box, {"left": 100, "top": 100, "width": 50, "height":
200});
        };
    };

    function buffer(obj, json) {
        // 1.1 清除定时器
        clearInterval(obj.timer);

        // 1.2 设置定时器
        var begin = 0;
        var target = 0;
        var speed = 0;
        obj.timer = setInterval(function () {
            // 1.3.0 旗帜
            var flag = true;
            for (var k in json) {
                // 1.3 获取初始值
                begin = parseInt(getCssStyleValue(obj, k)) || 0;
                target = json[k];
                // 1.4 求出步长
                speed = (target - begin) * .2;
                // 1.5 判断是否向上取整
```

```
                speed = target > begin ? Math.ceil(speed) : Math.floor(sp
eed);
                // 1.6 动起来
                obj.style[k] = begin + speed + "px";
                // 1.7 判断
                if (begin !== target) {
                    flag = false;
                }
            }
            // 1.3 清除定时器
            if (flag) {
                clearInterval(obj.timer);
            }
        }, 20);
    }
</script>
</body>
</html>
```

188. 封装缓动动画函数-回调(掌握)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <style>
        * {
            margin: 0;
            padding: 0;
        }

        #box {
            width: 100px;
            height: 100px;
            background-color: red;
            position: absolute;
        }
    </style>
</head>
<body>
<button id="btn">往右走</button>
<button id="btn1">往左走</button>
<div id="box"></div>
<script src="js/myfuncs.js"></script>
<script>
    window.onload = function () {
```

```
        // 0. 定义变量
        var box = $("box");

        // 1. 监听按钮的点击
        $("btn").onclick = function () {
            buffer(box, {"left": 800, "top": 120, "width": 230, "height":
 400}, function () {
                buffer(box, {"left": 100, "top": 100, "width": 50, "heigh
t": 200});
            });
        };
        $("btn1").onclick = function () {
            buffer(box, {"left": 100, "top": 100, "width": 50, "height":
200});
        };
    };

    function buffer(obj, json, fn) {
        // 1.1 清除定时器
        clearInterval(obj.timer);

        // 1.2 设置定时器
        var begin = 0;
        var target = 0;
        var speed = 0;
        obj.timer = setInterval(function () {
            // 1.3.0 旗帜
            var flag = true;
            for (var k in json) {
                // 1.3 获取初始值
                begin = parseInt(getCssStyleValue(obj, k)) || 0;
                target = json[k];
                // 1.4 求出步长
                speed = (target - begin) * .2;
                // 1.5 判断是否向上取整
                speed = target > begin ? Math.ceil(speed) : Math.floor(sp
eed);
                // 1.6 动起来
                obj.style[k] = begin + speed + "px";
                // 1.7 判断
                if (begin !== target) {
                    flag = false;
                }
            }
            // 1.3 清除定时器
            if (flag) {
                clearInterval(obj.timer);

                // 判断有没有回调函数
                if (fn) {
                    fn();
                }
```

```
            }
        }, 20);
    }
</script>
</body>
</html>
```

189. 封装缓动动画函数-透明度(掌握)

`C3` 透明度写法：`rgba(r, g, b, a)`
兼容旧浏览器写法：

```
opacity: .5;
filter: alpha(Opacity=50);
```

最终代码

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <style>
        * {
            margin: 0;
            padding: 0;
        }

        body{
            background-color: #000000;
        }

        #box {
            width: 100px;
            height: 100px;
            background-color: red;
            position: absolute;
            /*opacity: .5;*/
            /*filter: alpha(Opacity=50);*/
        }
    </style>
</head>
<body>
<button id="btn">往右走</button>
```

```
<button id="btn1">往左走</button>
<div id="box"></div>
<script src="js/myfuncs.js"></script>
<script>
    window.onload = function () {
        // 0. 定义变量
        var box = $("box");

        // 1. 监听按钮的点击
        $("btn").onclick = function () {
            buffer(box, {"opacity": .4, "left": 400, "top": 400, "width":
 250, "height": 300}, function () {
                buffer(box, {"left": 100, "top": 100, "width": 50, "heigh
t": 200, "opacity": 1});
            });
        };
        $("btn1").onclick = function () {
            buffer(box, {"left": 100, "top": 100, "width": 50, "height":
200});
        };
    };

    function buffer(obj, json, fn) {
        // 1.1 清除定时器
        clearInterval(obj.timer);

        // 1.2 设置定时器
        var begin = 0;
        var target = 0;
        var speed = 0;
        obj.timer = setInterval(function () {
            // 1.3.0 旗帜
            var flag = true;
            for (var k in json) {
                // 1.3 获取初始值
                if ("opacity" === k) {  // 设置透明度
                    begin = Math.round(parseFloat(getCssStyleValue(obj,
 k)) * 100) || 100;
                    target = parseInt(json[k] * 100);
                } else {  // 其他情况
                    begin = parseInt(getCssStyleValue(obj, k)) || 0;
                    target = json[k];
                }


                // 1.4 求出步长
                speed = (target - begin) * .2;
                // 1.5 判断是否向上取整
                speed = target > begin ? Math.ceil(speed) : Math.floor(sp
eed);
                // 1.6 动起来
                if ("opacity" === k) {
```

```
                    // 遵循 W3C的浏览器
                    obj.style.opacity = (begin + speed) / 100;
                    // ie 浏览器
                    obj.style.filter = "alpha(opacity:" + (begin + speed)
 + ")";
                } else {
                    obj.style[k] = begin + speed + "px";
                }

                // 1.7 判断
                if (begin !== target) {
                    flag = false;
                }
            }
            // 1.3 清除定时器
            if (flag) {
                clearInterval(obj.timer);

                // 判断有没有回调函数
                if (fn) {
                    fn();
                }

            }
        }, 20);
    }
</script>
</body>
</html>
```

190. 缓动案例-联动特效(掌握)


```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <style>
        * {
            margin: 0;
            padding: 0;
            border: none;
            list-style: none;
        }

        img {
            vertical-align: top;
        }

        #box{
            position: fixed;
            right: 0;
            bottom: 0;
        }

        #close{
```

```
            width: 25px;
            height: 25px;
            position: absolute; /* 绝对定位不占位置，找最近的定位元素做定位 */
            right: 0;
            top: 0;
        }
    </style>
</head>
<body>
<div id="box">
    <span id="close"></span>
    <div id="top">
        <img src="images/top.jpg" alt="">
    </div>
    <div id="bottom">
        <img src="images/bottom.jpg" alt="">
    </div>
</div>
<script src="js/myfuncs.js"></script>
<script>
    window.onload = function () {
        // 1. 获取需要的标签
        var box = $("box");
        var bottom = $("bottom");
        var top = $("top");
        var close = $("close");

        // 2. 监听点击
        close.onclick = function () {
            buffer(bottom, {"height": 0}, function () {
                buffer(box, {"width": 0}, function () {
                    close.style.display = "none";
                })
            })
        };
    }
</script>
</body>
</html>
```

191. 缓动案例-楼层特效-上(掌握)

> 快速生成标签 `ol>li{第$层}*5` + `tab`

处于选中状态的类老师说叫 样式类。
这个说法挺好的，应该要记住！

---

192. 缓动案例-楼层特效-中(掌握)

我目前对闭包的理解

```
for (var i = 0; i < arr.length; i++) {
    (function (index) {
        var item = arr[index];
        item.事件 = function () {
        };
    })(i);
}
```

- 首先闭包是位于一个 `for` 循环里面的。
- 第一步写出 `()(循环变量);`
- 第一个括号里面其实是一个函数，只不过是匿名函数的形式，把第二个括号里的 `循环变量` 作为实参，传递给第一个匿名函数的形参。
- `(匿名函数(形参))(循环变量);`
- **最终形式**：`(function (形参) { 匿名函数体; })(循环变量);`
  其中 `形参` 对应着 `循环变量`。

193. 缓动案例-楼层特效-下(掌握)

注意 `document.documentElement.scrollTop` 没有办法通过方括号获取，只能直接点语法拿。`scrollTop` 是一个数字，不是字符串，没有 `px` 后缀。

194. 缓动案例-楼层特效-结尾(掌握)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <style>
        * {
            margin: 0;
            padding: 0;
            list-style: none;
            border: none;
        }

        html, body, ul {
            width: 100%;
            height: 100%;
        }

        #ul li {
            width: 100%;
            height: 100%;
            text-align: center;
            font-size: 30px;
            color: #ffffff;
        }

        #ol {
            width: 80px;
            background-color: #cccccc;
            position: fixed;
            left: 50px;
            top: 200px;
            border: 1px solid #ffffff;
        }

        #ol li {
            text-align: center;
```

```
            line-height: 30px;
            border-bottom: 1px solid white;
            color: white;
            cursor: pointer;
        }

        #ol li.current {
            background-color: orangered;
        }

    </style>
</head>
<body>
<!--gps-->
<ol id="ol">
    <li class="current">第1层</li>
    <li>第2层</li>
    <li>第3层</li>
    <li>第4层</li>
    <li>第5层</li>
</ol>
<!--楼层-->
<ul id="ul">
    <li>第1层内容</li>
    <li>第2层内容</li>
    <li>第3层内容</li>
    <li>第4层内容</li>
    <li>第5层内容</li>
</ul>
<script src="js/myfuncs.js"></script>
<script>
    window.onload = function () {
        // 1. 获取需要的标签
        var ol = $("ol");
        var ul = $("ul");
        var ulLis = ul.children;
        var olLis = ol.children;
        var isClick = false;

        // 2. 上色
        var colorArr = ["brown", "green", "deepskyblue", "purple", "orang
e"];
        for (var i = 0; i < colorArr.length; i++) {
            ulLis[i].style.backgroundColor = colorArr[i];
        }

        // 3. 监听 gps 的点击
        for (var j = 0; j < olLis.length; j++) {
            (function (index) {
                var olLi = olLis[index];
                // 3.1 点击切换
                olLi.onmousedown = function () {
```

```
                    isClick = true;
                    // 排他，所以先清除样式
                    for (var i = 0; i < olLis.length; i++) {
                        olLis[i].className = "";
                    }
                    this.className = "current";

                    // 3.2 让内容滚动起来
                    buffer(document.documentElement, {"scrollTop": index
 * client().height}, function () {
                        isClick = false;
                    });
                };
            })(j);
        }

        // 4. 监听滚动
        window.onscroll = function () {
            if (!isClick) {
                // 4.1 获取滚动产生的高度
                var roll = Math.ceil(scroll().top);
                // 4.2 遍历
                for (var i = 0; i < olLis.length; i++) {
                    // 4.3 判断
                    if (roll >= ulLis[i].offsetTop) {
                        for (var j = 0; j < olLis.length; j++) {
                            olLis[j].className = "";
                        }
                        olLis[i].className = "current";
                    }
                }
            }
        }
    };
</script>
</body>
</html>
```

修改后的缓动动画

```
/**
 * 缓动动画
 * @param obj
 * @param json
 * @param fn
 */
function buffer(obj, json, fn) {
    // 1.1 清除定时器
```

```
    clearInterval(obj.timer);

    // 1.2 设置定时器
    var begin = 0;
    var target = 0;
    var speed = 0;
    obj.timer = setInterval(function () {
        // 1.3.0 旗帜
        var flag = true;
        for (var k in json) {
            // 1.3 获取初始值
            if ("opacity" === k) {  // 设置透明度
                begin = Math.round(parseFloat(getCssStyleValue(obj, k)) *
 100) || 100;
                target = parseInt(json[k] * 100);
            } else if ("scrollTop" === k) {
                begin = Math.ceil(obj.scrollTop);
                target = json[k];
            } else {  // 其他情况
                begin = parseInt(getCssStyleValue(obj, k)) || 0;
                target = json[k];
            }


            // 1.4 求出步长
            speed = (target - begin) * .2;
            // 1.5 判断是否向上取整
            speed = target > begin ? Math.ceil(speed) : Math.floor(spee
d);
            // 1.6 动起来
            if ("opacity" === k) {
                // 遵循 W3C的浏览器
                obj.style.opacity = (begin + speed) / 100;
                // ie 浏览器
                obj.style.filter = "alpha(opacity:" + (begin + speed) +
")";
            } else if ("scrollTop" === k) {
                obj.scrollTop = begin + speed;
            } else {
                obj.style[k] = begin + speed + "px";
            }

            // 1.7 判断
            if (begin !== target) {
                flag = false;
            }
        }
        // 1.3 清除定时器
        if (flag) {
            clearInterval(obj.timer);

            // 判断有没有回调函数
```

```
            if (fn) {
                fn();
            }

        }
    }, 20);
```

195. 缓动案例-音乐导航-上(掌握)

196. 缓动案例-音乐导航-下(掌握)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <style>
        *{
            margin: 0;
            padding: 0;
            list-style: none;
            border: none;
        }
        a{
            text-decoration: none;
            color: #000000;
        }
        #nav{
            width: 900px;
            height: 40px;
            border: 1px solid #cccccc;
            margin: 100px auto;
            overflow: hidden;
        }
        #nav ul{
            width: 910px;
        }
        #nav ul li{
            float: left;
            width: 100px;
```

```
            line-height: 40px;
            text-align: center;
            background: url("images/_r1_c1.png") no-repeat 0 0;
            border-right: 1px dashed #cccccc;
            position: relative;
        }
        #nav ul li a{
            width: 100%;
            height: 100%;
            display: inline-block;
        }
        span{
            width: 100px;
            height: 40px;
            background-color: skyblue;
            position: absolute;
            left: 0;
            top: 40px;
            z-index: -1;
        }

    </style>
</head>
<body>
    <nav id="nav">
        <ul id="ul">
            <li><a href="">首页</a><span></span><audio
src="source/a1.mp3"></audio></li>
            <li><a href="">新头条</a><span></span><audio src="source/a2.mp
3"></audio></li>
            <li><a href="">电视剧</a><span></span><audio src="source/a3.mp
3"></audio></li>
            <li><a href="">新电影</a><span></span><audio src="source/a4.mp
3"></audio></li>
            <li><a href="">小游戏</a><span></span><audio src="source/a5.mp
3"></audio></li>
            <li><a href="">小说汇</a><span></span><audio src="source/a6.mp
3"></audio></li>
            <li><a href="">旅游假</a><span></span><audio src="source/a7.mp
3"></audio></li>
            <li><a href="">正品购</a><span></span><audio src="source/a8.mp
3"></audio></li>
            <li><a href="">今日团</a><span></span><audio src="source/a9.mp
3"></audio></li>
        </ul>
    </nav>
    <script src="js/myfuncs.js"></script>
    <script>
        window.onload = function () {
            // 1. 获取标签
            var ul = $("ul");
            var allLis = ul.children;
```

```
            // 2. 改变背景
            for (var i = 0; i < allLis.length; i++) {
                allLis[i].style.backgroundPosition = "0 " + i * (-40) +
"px";
                // 2.1 监听鼠标进入
                allLis[i].onmouseover = function () {
                    // 2.1.1 缓动动画
                    buffer(this.children[1], {top: 0});
                    // 2.1.2 播放音乐
                    this.children[2].currentTime = 0;
                    this.children[2].play();
                }
                // 2.2 监听鼠标离开
                allLis[i].onmouseout = function () {
                    buffer(this.children[1], {top: 40});
                };
            }
            // 3. 监听键盘的点击
            document.onkeydown = function (event) {
                var e = event || window.event;
                // 3.1 获取键的编码
                var keyCode = e.keyCode - 49;  // 1 是 49
                buffer(allLis[keyCode].children[1], {top: 0}, function ()
 {
                    buffer(allLis[keyCode].children[1], {top: 40});
                });
                // 3.2 播放音乐
                allLis[keyCode].currentTime = 0;
                allLis[keyCode].children[2].play();
            };
        };
    </script>
</body>
</html>
```

197. 网易轮播图-上(掌握)

类名用中划线，`id` 用下划线。

198. 网易轮播图-样式-1(掌握)

199. 网易轮播图-动态效果-1(掌握)

200. 网易轮播图-动态效果-1(掌握)

201. 网易轮播图-动态效果-2(掌握)

202. 网易轮播图-动态效果-3(掌握)

203. 网易轮播图-动态效果-4(掌握)

`index.html`

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <link rel="stylesheet" href="css/index.css">
</head>
```

```
<body>
<div id="slider" class="slider">
    <div id="slider_scroll" class="slider-scroll">
        <div class="slider-main" id="slider_main">
            <div class="slider-main-img">
                <a href="">
                    <img src="images/1.jpeg" alt="">
                </a>
            </div>
            <div class="slider-main-img">
                <a href="">
                    <img src="images/2.jpeg" alt="">
                </a>
            </div>
            <div class="slider-main-img">
                <a href="">
                    <img src="images/3.jpeg" alt="">
                </a>
            </div>
            <div class="slider-main-img">
                <a href="">
                    <img src="images/4.jpeg" alt="">
                </a>
            </div>
            <div class="slider-main-img">
                <a href="">
                    <img src="images/5.jpeg" alt="">
                </a>
            </div>
            <div class="slider-main-img">
                <a href="">
                    <img src="images/6.jpeg" alt="">
                </a>
            </div>
        </div>
    </div>
    <div class="slider-ctl" id="slider_ctl">
        <span class="slider-ctl-prev"></span>
        <span class="slider-ctl-next"></span>
    </div>
</div>
<script src="js/myfuncs.js"></script>
<script src="js/index.js"></script>
</body>
</html>
```

`index.css`

```css
*{
    margin: 0;
    padding: 0;
    border: none;
}

img{
    vertical-align: top;
}

.slider{
    width: 310px;
    height: 260px;
    position: relative;
    margin: 100px auto;
    overflow: hidden;
}

.slider-scroll{
    width: 310px;
    height: 220px;
    position: relative;
}

.slider-main{
    width: 620px;
    height: 220px;
    background-color: purple;
}

.slider-main-img{
    width: 310px;
    height: 220px;
    position: absolute;
}

.slider-main-img img{
    width: 100%;
    height: 100%;
}

.slider-ctl{
    cusor: pointer;
    text-align: center;
}

.slider-ctl-prev,.slider-ctl-next{
    background: url("../images/icon.png");
    width: 20px;
    height: 34px;
    position: absolute;
```

```
        top: 50%;
        margin-top: -35px;
        cursor: pointer;
    }

    .slider-ctl-prev{
        left: 5px;
    }

    .slider-ctl-next{
        background-position: -9px -45px;
        right: 5px;
    }

    .slider-ctl-icon{
        width: 24px;
        height: 5px;
        background: url("../images/icon.png") no-repeat -24px -790px;
        display: inline-block;
        margin: 5px;
        cursor: pointer;
        text-indent: 20em;
    }

    .current{
        background-position: 0 -770px;
    }
```

index.js

```
    window.onload = function () {
        // 1. 获取需要的标签
        var slider = $("slider");
        // slider_main 就是有两个图片宽度的，可以滚动的那个东西
        // 它下面有所有的图
        var slider_main = $("slider_main");
        var slider_main_img = slider_main.children;
        var slider_ctl = slider.children[1];
        var iNow = 0;  // 当前索引

        // 2. 动态创建指示器
        var span;
        for (var i = 0; i < slider_main_img.length; i++) {
            span = document.createElement("span");
            span.className = "slider-ctl-icon";
            span.innerText = slider_main_img.length - i - 1;
            slider_ctl.insertBefore(span, slider_ctl.children[1]);
        }
```

```
// 3. 让第一选中：注意赋值是覆盖操作，不是叠加操作！
slider_ctl.children[1].className = "slider-ctl-icon current";

// 4. 让滚动的内容归位
var scroll_w = slider.offsetWidth;
for (var j = 1; j < slider_main_img.length; j++) {
    slider_main_img[j].style.left = scroll_w + "px";
}

// 5. 遍历监听操作
var slider_ctl_children = slider_ctl.children;
for (var i = 0; i < slider_ctl_children.length; i++) {
    // 5.1 监听点击
    slider_ctl_children[i].onmousedown = function () {
        // 5.2 判断点击的位置
        if (this.className === "slider-ctl-prev") {  // 左边
            // 注意：点左边是看上一张，图片要右移！！！
            /*
             1. 当前可视区域的图片快速右移
             2. 上一张图片快速出现在可视区域的左边
             3. 让这张图片做动画进入
             */
            buffer(slider_main_img[iNow], {"left": scroll_w});
            iNow--;
            // 判断
            if (iNow < 0) {
                iNow = slider_main_img.length - 1;
            }
            slider_main_img[iNow].style.left = -scroll_w + "px";
            buffer(slider_main_img[iNow], {"left": 0});

        } else if (this.className === "slider-ctl-next") {  // 右边
            /*
             1. 当前可视区域的图片快速左移
             2. 下一张图片快速出现在可视区域的右边
             3. 让这张图片做动画进入
             */
            buffer(slider_main_img[iNow], {"left": -scroll_w});
            iNow++;
            // 判断
            if (iNow > slider_main_img.length - 1) {
                iNow = 0;
            }
            slider_main_img[iNow].style.left = scroll_w + "px";
            buffer(slider_main_img[iNow], {"left": 0});
        } else {  // 下边
            /*
             1. 用当前点击的索引和选中索引对比
             2. 点击的 > 选中的，相当于点击了右边的按钮
             3. 点击的 < 选中的，相当于点击了左边的按钮
             */
```

```
                // 获取索引号
                var index = parseInt(this.innerText);
                // 对比
                if (index > iNow) {
                    buffer(slider_main_img[iNow], {"left": -scroll_w});
                    slider_main_img[index].style.left = scroll_w + "px";
                } else if (index < iNow) {
                    buffer(slider_main_img[iNow], {"left": scroll_w});
                    slider_main_img[index].style.left = -scroll_w + "px";
                }
                iNow = index;
                buffer(slider_main_img[iNow], {"left": 0});
            }
            changeIndex();
        };
    }

    // 6. 切换索引
    function changeIndex() {
        for (var i = 1; i < slider_ctl_children.length - 1; i++) {
            slider_ctl_children[i].className = "slider-ctl-icon";
        }
        slider_ctl_children[iNow + 1].className = "slider-ctl-icon curren
t";
    }

    // 7. 自动播放
    function autoplay() {
        /*
         1. 当前可视区域的图片快速左移
         2. 下一张图片快速出现在可视区域的右边
         3. 让这张图片做动画进入
         */
        buffer(slider_main_img[iNow], {"left": -scroll_w});
        iNow++;
        // 判断
        if (iNow > slider_main_img.length - 1) {
            iNow = 0;
        }
        slider_main_img[iNow].style.left = scroll_w + "px";
        buffer(slider_main_img[iNow], {"left": 0});
        changeIndex();
    }

    var timer = setInterval(autoplay, 1000);

    // 8. 设置和清除定时器
    slider.onmouseover = function () {
        clearInterval(timer);
    };
    slider.onmouseout = function () {
        timer = setInterval(autoplay, 1000);
```

```
    };
};
```

204. 旋转木马特效-前奏(掌握)

一、数组的常见操作

> pop() 删除最后一个
>
> shift() 删除第一个
>
> push() 追加 添加到最后面
>
> unshift() 添加到 第一个位置

> 个人理解：数组就像是一个栈结构，数组的开头为栈底，数组的末尾为栈顶。
> `push` 和 `pop` 都是栈的相关方法，只从栈顶操作元素。
> 而 `shift` 在英语中有位移的意思，人的阅读习惯一般是从左到右，所以 `shift` 相当于把数组的头指针向右移动一位，也就是删除了第一个元素。
> `unshift` 就是把头指针向左移动，并添加一个元素进去。

205. 旋转木马特效-界面布局(掌握)

206. 旋转木马特效-动态特效-上(掌握)

207. 旋转木马特效-动态特效-下(掌握)

效果图

改进后的缓动动画

主要是增加了对 `opacity` 的直接设置，以及修正了透明度闪烁的问题（这个我还不太明白是怎么回事）。

```
/**
 * 缓动动画
 * @param obj
 * @param json
 * @param fn
 */
function buffer(obj, json, fn) {
    // 1.1 清除定时器
    clearInterval(obj.timer);

    // 1.2 设置定时器
    var begin = 0;
    var target = 0;
    var speed = 0;
    obj.timer = setInterval(function () {
        // 1.3.0 旗帜
        var flag = true;
        for (var k in json) {
            // 1.3 获取初始值
            if ("opacity" === k) {  // 设置透明度
                begin = parseInt(parseFloat(getCssStyleValue(obj, k)) * 1
00);
                target = parseInt(parseFloat(json[k]) * 100);
            } else if ("scrollTop" === k) {
                begin = Math.ceil(obj.scrollTop);
                target = parseInt(json[k]);
            } else {  // 其他情况
                begin = parseInt(getCssStyleValue(obj, k)) || 0;
                target = parseInt(json[k]);
            }


            // 1.4 求出步长
            speed = (target - begin) * .2;
            // 1.5 判断是否向上取整
```

```
            speed = target > begin ? Math.ceil(speed) : Math.floor(spee
d);
            // 1.6 动起来
            if ("opacity" === k) {
                // 遵循 W3C的浏览器
                obj.style.opacity = (begin + speed) / 100;
                // ie 浏览器
                obj.style.filter = "alpha(opacity:" + (begin + speed) +
")";
            } else if ("scrollTop" === k) {
                obj.scrollTop = begin + speed;
            } else if ("zIndex" === k) {
                obj.style[k] = json[k];
            } else {
                obj.style[k] = begin + speed + "px";
            }

            // 1.7 判断
            if (begin !== target) {
                flag = false;
            }
        }
        // 1.3 清除定时器
        if (flag) {
            clearInterval(obj.timer);

            // 判断有没有回调函数
            if (fn) {
                fn();
            }

        }
    }, 20);
}
```

最终代码

`index.html`

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <style>
        *{
            margin: 0;
            padding: 0;
            border: none;
            list-style: none;
        }
```

```
        img{
            vertical-align: top;
        }
        body{
            background-color: darkgray;
        }
        #slider{
            width: 1200px;
            height: 460px;
            margin: 100px auto;
            position: relative;
            /*background-color: red;*/
        }
        #slider li{
            position: absolute;
            left: 200px;
            top: 0;
        }
        #slider li img{
            width: 100%;
            height: 100%;
        }

        #slider_ctrl{
            opacity: 0;
        }
        .slider-ctrl-prev,.slider-ctrl-next{
            width: 76px;
            height: 112px;
            position: absolute;
            top: 50%;
            margin-top: -56px;
            z-index: 9;
        }
        .slider-ctrl-prev{
            background: url("images/prev.png") no-repeat;
            left: 0;
        }
        .slider-ctrl-next{
            background: url("images/next.png") no-repeat;
            right: 0;
        }
    </style>
</head>
<body>
    <div id="slider">
        <ul id="slider_main">
            <li><img src="images/slidepic1.jpg" alt=""></li>
            <li><img src="images/slidepic2.jpg" alt=""></li>
            <li><img src="images/slidepic3.jpg" alt=""></li>
            <li><img src="images/slidepic4.jpg" alt=""></li>
            <li><img src="images/slidepic5.jpg" alt=""></li>
```

```
        </ul>
        <div id="slider_ctrl">
            <span class="slider-ctrl-prev"></span>
            <span class="slider-ctrl-next"></span>
        </div>
    </div>
    <script src="js/myfuncs.js"></script>
    <script>
        window.onload = function () {
            // 1. 获取需要的标签
            var slider = $("slider");
            var slider_main = $("slider_main");
            var allLis = slider_main.children;
            var slider_ctrl = $("slider_ctrl");

            // 2. 设置指示器的显示和隐藏
            slider.onmouseover = function () {
                buffer(slider_ctrl, {"opacity": 1});
            };
            slider.onmouseout = function () {
                buffer(slider_ctrl, {"opacity": 0});
            };

            // 3. 定位
            var json = [
                {  // 1
                    "width": 400,
                    "top": 20,
                    "left": 50,
                    "opacity": .2,
                    "zIndex": 2
                },
                {  // 2
                    "width": 600,
                    "top": 70,
                    "left": 0,
                    "opacity": .8,
                    "zIndex": 3
                },
                {  // 3
                    "width": 800,
                    "top": 100,
                    "left": 200,
                    "opacity": 1,
                    "zIndex": 4
                },
                {  // 4
                    "width": 600,
                    "top": 70,
                    "left": 600,
                    "opacity": .8,
                    "zIndex": 3
```

```
                },
                {  // 5
                    "width": 400,
                    "top": 20,
                    "left": 750,
                    "opacity": .2,
                    "zIndex": 2
                }
            ];
            for(var i = 0; i < json.length; i++) {
                buffer(allLis[i], json[i])
            }

            // 4. 监听点击
            for (var j = 0; j < slider_ctrl.children.length; j++) {
                var item = slider_ctrl.children[j];
                item.onmousedown = function () {
                    if (this.className === "slider-ctrl-prev") {  // 左边
                        json.push(json.shift());

                    } else {  // 右边
                        json.unshift(json.pop());
                    }
                    // 重新布局
                    for(var i = 0; i < json.length; i++) {
                        buffer(allLis[i], json[i])
                    }
                };
            }
        };
    </script>
</body>
</html>
```

208. 面向对象-知识点回顾

二、面向对象

对象是什么？

对象就是带有属性和方法的数据类型！

任何一门高级语言都要面向对象，JavaScript则是基于原型的面向对象语言，因此，我们的思维要由**面向过程**转向**面向对象**：

面向过程	面向对象
onload（初始化整个程序）	构造函数（初始化对象）
全局变量	属性（对象.属性）　this.属性
函数	方法（构造函数.原型.方法）　构造函数.prototype.方法

面向对象：

面向对象里面：类和对象

类是对象的抽象，而对象是类的具体实例

一切事物皆对象 JavaScript 一切皆对象

面向对象的特性：

- 抽象
- ◦ 抽象就是忽略一个主题中与当前目标无关的那些方面，以便更充分地注意与当前目标有关的方面。
- 封装
- ◦ 封装是把过程和数据封闭起来，对数据的访问只能通过开放的接口。
- 继承
- ◦ 子类对象继承使用父类的属性和方法。
- 多态
- ◦ 多态是指两个或多个属于不同类的对象，对于同一个消息（方法调用）作出不同响应的方式。

2.1 构造函数

> 所有的构造函数有一个特点：首字母大写；
>
> 在js中我们可以理解为只要执行以后能够返回新的对象的函数就是构造函数。

构造函数技巧的最大目的：创造完全独立的对象，互相之间不影响。

2.2 关键词new

- 将一个函数变成对象并返回
- 在这个函数的内部将this指向函数本身。
- new这个关键词实际上能够将任何函数直接变成一个对象。它只有在和构造函数配合的时候才有用，它相当于可以化简构造函数自己创造对象和返回对象的步骤。

> 这里不是很懂？？？
> 老师说：new 加在函数前面，函数就变成了对象？？？
> 估计说法有误

2.3 构造器(constructor) 和 原型属性 (prototype)

在任何一个对象中都有构造器和原型属性，包括原生的对象，比如： Date, Array等；

- **constructor 返回对创建此对象的 构造函数的引用**
- **prototype 让我们有能力向对象添加属性和方法**

  > prototype它的作用就是构造函数的一个共享库；在这个共享库里面存储的所有数据将来都会被所有的新对象公用。 这样大大降低了创建方法的成本。

- **原型共享库是谁使用的？**
  - 构造函数使用原型库，所有将来的对象共享这个原型库。
  - 如果把方法都写在构造函数的原型库里面，将来还可以通过原型继续拓展。 、
- **原型的工作原理？**
-
  - 在网页发布以后，原型的工作会自动做以下两件事情：

```
第一：自动将原型库中的所有内容都放在将来的对象身上；
第二：如果共享库中的内容发生变化会自动更新所有对象上的数据。
```

- **注意：**
  - 在面向对象的写法当中，原型的共享库里面所有的方法中的this默认情况都会指向将来的对象。
  - 只有在两个情况会发生变化，那么这两个情况一定要检查作用域：

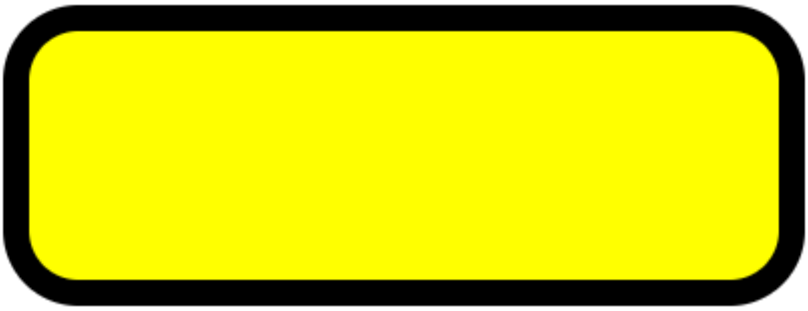
```
第一：如果在事件的作用域中，this的指向会变成事件源。
第二：如果在定时器的作用域中，this的指向会变成window。
```

209. 面向对象-构造函数-回顾

```
<script>
```

```
    function Dog(option) {
        this._init(option);
    }

    Dog.prototype = {
        // 属性
        _init: function (option) {
            this.name = option.name;
            this.age = option.age;
            this.dogFriends = option.dogFriends;
        },

        // 方法
        eat: function (something) {
            console.log(this.name + "吃" + something);
        },
        run: function (somewhere) {
            console.log(this.name + "跑" + somewhere);
        }
    };

    // 实例化
    var sDog = new Dog({name: "小花", age: 1});
    console.log(sDog);
    console.log(typeof sDog);
    console.log(sDog.name);
    console.log(sDog.age);
    console.log(sDog.dogFriends);
    sDog.eat("奶");
    sDog.run("操场");

    var bDog = new Dog({name: "小花", age: 10, dogFriends: ["大大", "小
小"]});
    console.log(bDog);
    console.log(sDog === bDog);  // fasle
    console.log(sDog.eat === bDog.eat);  // true

</script>
```

210. 面向对象-矩形



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<div id="main"></div>
<script>
    function Rect(option) {
        this._init(option);
    }

    Rect.prototype = {
        _init: function (option) {
            option = option || {};

            // 父标签的id
            this.parentId = option.parentId;
            // 位置
            this.left = option.left || 0;
            this.top = option.top || 0;
            // 自身属性
            this.width = option.width || 100;
            this.height = option.height || 50;
            this.backgroundColor = option.backgroundColor || "blue";
            this.border = option.border || 0;
            this.borderRadius = option.borderRadius || 0;
        },

        render: function() {
            var parentNode = document.getElementById(this.parentId);
            var childNode = document.createElement("div");
            childNode.style.position = "absolute";
            childNode.style.left = this.left + "px";
            childNode.style.top = this.top + "px";
            childNode.style.width = this.width + "px";
            childNode.style.height = this.height + "px";
            childNode.style.backgroundColor = this.backgroundColor;
            childNode.style.border = this.border;
            childNode.style.borderRadius = this.borderRadius + "px";
```

```
                parentNode.appendChild(childNode);
            }
        }

        // 实例化
        var rect = new Rect({
            parentId: "main",
            left: 100,
            top: 200,
            width: 300,
            height: 100,
            backgroundColor: "yellow",
            border: "10px solid #000",
            borderRadius: 30
        });
        rect.render();
    </script>
    </body>
    </html>
```

完成于 `2019.3.8`