目录

1.认识运动

- 动画 图像
  - 原理：人眼能够识别的**最小时间间隔**是 `18` 帧。
  - 动画：只需要让动画的切换时间间隔大于 `18` 帧，一般情况下电影院里放映的电影是 `24` 帧。
- 简单的运动（匀速运动），让一个div从左到右运动

```
// 完美运行的代码
var oBtn = document.getElementById("btn1");
var oDiv = document.getElementById("div1");

var speed = 7;
var timear = null
oBtn.onclick = function(){
  clearInterval(timear);
  timear = setInterval( () => {
    if(oDiv.offsetLeft >= 500){
      clearInterval(timear);
    }else{
      oDiv.style.left = oDiv.offsetLeft + speed + "px";
    }
  },30 )
}
```

  - 遇到的问题
    1. 停不下来
    2. 当速度取某些值停不下来
    3. 到达目的值以后，点击还会继续往前运动。
       - `if...else` 将运动和停止分开
    4. 重复点击按钮，速度加快
       - 保证只有一个定时器在启动，每次启动定时器之前，先将上一次定时器关闭
- 运动框架：写动画必写的代码
  1. `if...else` 将运动和停止分开
  2. 每次启动定时器之前，先将上一次定时器关闭

2.分享到和淡入淡出案例

1. 案例- `分享到` 菜单

    - 通过 移入移出 执行封装好的函数
    - 如果实现的功能有很多重复的代码，要封装成一个函数，不同的地方通过传参来修改。
2. 淡入淡出效果
    - 注意使用中间变量
        - 例如透明度的兼容 `opacity: 0.3;` / `filter: alpha(opacity=30);`
            - 这时，两个都不好取值，用自定义变量 `var alpha = 30;`，赋值时两个透明度都设置

3.缓冲运动

- 类似于刹车：
    - 不是物理规律的特点：速度和距离成正相关。
- 速度 = 到目标剩余距离 / 固定值;
    - 这样的速度，只会越来越小。无法达到指定位置。
        - 计算机不会计算小数点像素，遇到小数点像素，当做`0`处理，根据从左向右 还是 从右向左运动，来进行向上或向下取整。

```
speed = speed > 0 ? Math.ceil(speed) : Math.floor(speed);
```

4.缓冲运动案例-缓冲菜单

- 获取浏览器窗口的宽和高： `window.innerHeight` `window.innerWidth`

```
var speed = (iTarget - iCur) / 8; (经过长期试验，/8是动画最合理的)
speed = speed > 0 ? Math.ceil(speed) : Math.floor(speed);
```

    - `iTarget` ： 菜单目的值
    - `iCur`： 当前菜单的位置
- 缓冲菜单实现原理：
  目标`top`：(`window.offsetHeight` - `本节点.offsetHeight`) / `2` + `父节点.scrollTop`;
    - (`可见窗口高` - `本身高`) / `2` + `父节点已经滑动的隐藏高度`
        - 进行缓冲运动: `top` - `本节点.offsetTop` / `8`;

5.多物体运动

- 让谁进行运动不确定， `node`参数,传入要运动的节点
- 多物体运动，实际上只有一个定时器，**多物体不能共用一个定时器。**
- 让**每一个运动的物体都有一个定时器。**
    - 给每个事件元素 都添加一个 `this.timer = null`;

- 多物体淡入淡出
    - 不能共用一个中间变量`alpha`, 在事件函数之前，给事件函数元素添加`alpha`属性

- 多物体运动

1. 定时器不能共用
2. 任何数据都不能共用

6.offset系列的问题

- `offset` / `client` / `scroll`
  - 在进行取值后再赋值的问题上，要根据实际情况使用

```
div.style.width = div.clientWidth + 10 + "px";
```

- `offsetWidth`： `width` + `padding` + `border` (包括滚动条)
- `clientWidth`： `width` + `padding` (不包括滚动条)
- `scrollWidth`： `width` + `padding` (超出宽度值计算一边`padding`) 对象内容实际宽度，包括隐藏宽度

- 如果要**在元素当前样式基础上进行操作**，尽量**使用** `node.getcurrentStyle[] || getComputedStyle(node)[]`
  - 因为获取的是书写的属性值
  - **元素包含滚动条时，元素实际的宽度**(`width`)就**会缩小。**
    - `node.getcurrentStyle[] || getComputedStyle(node)[]` 如果节点包含滚动条，获取的是非缩小的宽度

7.多物体多样式运动

- 每个物体运动变化的属性不一样
- 函数要传入的参数就变成 `(节点，属性名，目标值)`

- 处理透明度问题
  1. 取值的时候不一样 (自动取整对透明度没有过渡)
  2. 赋值的时候不一样 (没有px单位)
  - 解决方案：在取值的时候进行一次判断，`属性名 == opacity` 就进行浮点数取值 * 100 取整；在赋值时，进行判断 `属性名 == opacity` 进行两种方法赋值。

8.链式运动(回调函数)

- 链式运动：在第一个动画结束的时候，开始第二个动画。
  - 关键点，找到第一个动画结束的时候。

- 回调函数：我们**把函数当参数传入**，并且**在合适的地方调用的方式**，**叫回调函数。**在别的语言(C语言、C++)叫做 函数指针。
- 链式运动就是在动画结束的时候，通过回调函数，执行下一部分动画

```
aDivs[0].onmouseover = function(){
  // 先执行width变300，执行完成后，在clearInterval()后 调用传入的函数
  startMove(this, "width", 300,  function(){
    // 执行完成后，在clearInterval()后 调用传入的函数
    startMove(this, "height", 300, function(){
      // 执行完成后，没有传入函数，结束运动
      startMove(this, "opacity", 100);
    })
  });
}
```

9.打砖块

1. 球在父容器内的弹动
2. 平台 X轴的位移
3. 砖块的插入 颜色随机
4. 碰撞检测
   - 逆向思维：两个物体怎么样才绝对碰不上
     - `物体1` 和 `物体2` 是两个div
     - `物体1` 的上边 小于 `物体2` 的下边 `t1 < b2`
     - `物体1` 的右边 小于 `物体2` 的左边 `r1 < l2`
     - `物体1` 的下边 大于 `物体2` 的上边 `b1 > t2`
     - `物体1` 的左边 大于 `物体2` 的右边 `l1 > r2`
       - 用 或运算符 `||` 只要一个为假，就是发生了碰撞
5. 砖块文档流转定位

```
* `float` 转 `position`
  * 不能在循环获取坐标时 给元素添加`position : absolute`; 这样的话，`float`就会失效，后面的`float`

  * 当`float`和`position`混用时，只能指定`static`和`relative`，如果指定了`absolute`或者`fixed`，`
```

10.圆周运动

- 画圆的步骤： (圆规画圆)
  1. 确定圆心的位置
  2. 确定半径
  3. 旋转 顺时针(正方向)，逆时针

  - `Math.PI` = 1π(弧度) = 180°
  - 1角度 = `Math.PI` / 180;

  - sin(圆的已运动的角度) = 对边 / 斜边
  - 对边(x轴距离) = sin(圆的已运动的角度) * 斜边

- 【注】要记录已经旋转的角度

```
var i = 0;
i++;
radina = i * Math.PI / 180;
```

- 关于正弦、余弦、正切象限问题

	一	二	三	四
正弦(sin):	正	正	负	负
余弦(cos):	正	负	负	正
正切(tan):	正	负	正	负

11.完美运动框架

- 链式运动(回调函数)基础上改造

- 宽高透明度同时发生变化
  - 遇到的问题：每一个元素节点只有一个定时器，当同时调用两次函数时，第一次被调用的函数的定时器就会被立马关闭。
  - 用一个函数给div同时设置 宽 高 背景
    - 我们将需要传入的多个样式，变成一个对象，传入对象，函数遍历对象就ok
    - 参数变对象，定时器里写循环 遇到的问题：
      - 当一个属性到达目的值，就会关闭这个定时器。
      - 解决方法：把关闭定时器的判断放到定时器外，设置一个 `中间变量 = true`，当有一个值没到达目的值时，`中间变量 = false`，只有所有属性到达目的值，才会关闭定时器。
        - 这种方法和自己的每个属性都有一个定时器更好，命名不会冲突，更节省空间。

```
window.onload = function(){
  var  oDiv1 = document.getElementById("div1");
  /*
      宽高透明度同时发生变化
  */
  oDiv1.onmouseover = function(){
    startMove(this,{
      width: 300,
      height: 300,
      opacity: 100
    },function(){
      alert("动画结束了")
    });
    // startMove(this,"height",300);
  }
  oDiv1.onmouseout = function(){
    startMove(this,{
      width: 100,
      height: 100,
      opacity: 30
    });
  }
}
/*
    回调函数：我们把函数当参数传入，并且在合适的地方调用的方式，叫回调函数。

    遇到的问题：
        当一个属性到达目的值，就会关闭这个定时器
*/
<div STYLE="page-break-after: always;"></div>
function startMove(node, cssObj, complete){ //complete = show
  clearInterval(node.timer);
  node.timer = setInterval( function(){
    //判断属性到达目的值没
    var isEnd = true;
    for(var attr in cssObj){
      var iTarget = cssObj[attr];
      // 计算速度
      var iCur = null;
      if(attr == "opacity"){
        iCur = parseInt( parseFloat( getStyle(node, attr) ) * 100);
      }else{
        iCur = parseInt(getStyle(node, attr));
      }
      console.log(iCur);
      var speed = (iTarget - iCur) / 8;
      speed = speed > 0 ? Math.ceil(speed) : Math.floor(speed);

      iCur += speed;
      if(attr == "opacity"){
        node.style[attr] = iCur / 100;
        node.style.filter = "alpha(opacity = " + iCur + ")";
      }else{
        node.style[attr] = iCur + "px";
```

```
      }
      // 如果属性没到达目的值 isEnd = false
      if(iCur != iTarget){
        isEnd = false;
      }
    }
    //当isEnd为true时，说明所有的属性已经到达目的值
    if(isEnd){
        if(complete){
            complete.call(node);
        }
        clearInterval(node.timer);
    }
  },30 );
}

//获取浏览器有效样式的兼容写法
function getStyle(node, styleStr){
    return node.getcurrentStyle ? node.getcurrentStyle[styleStr] :
    getComputedStyle(node)[styleStr];
}
```

12.多图片的缩放

- 九宫格布局：3 x 3 防止误触

1. 布局的时候：**相对定位**
    - 实际放大的时候：**必须是绝对定位** （如果还是浮动，就会挤开后面的元素）
    - 文档流的转换: 相对定位 => 绝对定位
2. 中心放大 (放大的时候最好不要改变元素的坐标)
    - 使元素的`margin-left`和`margin-top` = 放大像素的一半

13.banner图效果

- 四张轮播图循环播放
    1. 复制一份，形成8张图片，然后重新设置图片父容器ul的宽
    2. 当`ul`的`offsetLetf` = `ul.offsetWidth / 2` 的时候， `ul.style.left = "0px"`;