[笔记][Bootstrap][05-Javascript动画库]

前端

---

39. GSAP动画插件

- `GSAP` 是通过 `js` 来实现动画，而 `animate` 是通过 `CSS` 来实现动画。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>01-GSAP开篇</title>
    <script src="js/TweenMax.js"></script>
    <style>
        *{
            margin: 0;
            padding: 0;
```

```
        }
        div{
            width: 100px;
            height: 100px;
            background: red;
        }
    </style>
</head>
<body>
<div class="box"></div>
<script>
    /*
    1.什么是ScrollMagic?
    ScrollMagic是一个滚动视差插件
    ScrollMagic本身比较简单，只包含2个类:
    crollMagic.Controller 一个控制器类，用于总体的调度 ；
    ScrollMagic.Scene 一个场景类，用于设计具体的变换。

    需要注意的是，它本身并没有集成 animation的控制方法，动画的实现，需要引入插件 GSAP 或者是 Velocity
    * */
    /*
    1.什么是GSAP?
    GSAP(GreenSock Animation Platform)是一个从flash时代一直发展到今天的专业动画库

    2.GSAP优点
    1、速度快。GSAP专门优化了动画性能，使之实现和CSS一样的高性能动画效果。
    2、轻量与模块化。模块化与插件式的结构保持了核心引擎的轻量，TweenLite包非常小（基本上低于7kb）。GSAP提供了TweenLite，TimelineLite，TimelineMax 和 TweenMax不同功能的动画模块，你可以按需使用。
    3、没有依赖。
    4、灵活控制。不用受限于线性序列，可以重叠动画序列，你可以通过精确时间控制，灵活地使用最少的代码实现动画。

    3.GSAP版本
    GSAP提供4个库文件供用户使用
    1.TweenLite：这是GSAP动画平台的核心部分，使用它可以用来实现大部分的动画效果，适合来实现一些元素的简单动画效果。
```

2.TimelineLite：一个强大的，轻量级的序列工具，它就如一个存放补间动画的容器，可以很容易的整体控制补间动画，且精确管理补间动画彼此之间的时间关系。比如动画的各种状态，Pause，reverse，restart，speed up，slow down，seek time，add labels等。它适合来实现一些复杂的动画效果。

3.TimelineMax：扩展TimelineLite，提供完全相同的功能再加上有用的（但非必需）功能，如repeat，repeatDelay，yoyo，currentLabel()等。TimelineMax的目标是成为最终的全功能工具，而不是轻量级的。

4.TweenMax：可以完成TimelineLite做的每一件事，并附加非必要的功能，如repeat，yoyo，repeatDelay(重复延迟)等。它也包括许多常见的插件，如CSSPlugin，这样您就不需自行载入多个文件。侧重于全功能的，而不是轻量级的。

>>建议在开发中使用TweenMax这个全功能的js文件，它包括了GreenSock动画平台的所有核心的功能。

```
    官网地址：http://www.greensock.com/
    github地址：https://github.com/greensock/GreenSock-JS/
    中文网：https://www.tweenmax.com.cn/
    * */
    // 1.创建TweenMax对象
    /*
    第一个参数：需要执行动画的元素
    第二个参数：动画执行的时长
    第三个参数：具体执行动画的属性
    * */
    // new TweenMax(".box", 3, {x: 500});
    // 2.利用静态方法执行动画
    // 从当前位置到指定位置
    // TweenMax.to(".box", 3, {x: 500});
    // 从指定位置到当前位置
    // TweenMax.from(".box", 3, {x: 500});
    // 从第一个指定的位置到第二个指定的位置
    TweenMax.fromTo(".box", 3, {x: 500}, {x: 200});
</script>
</body>
</html>
```

40. GSAP交叉动画

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>02-GSAP交叉动画</title>
    <script src="js/TweenMax.js"></script>
    <style>
        *{
            margin: 0;
            padding: 0;
        }
        div{
            width: 100px;
            height: 100px;
            background: red;
        }
        .box2{
            background: blue;
        }
        .box3{
            background: green;
        }
    </style>
</head>
<body>
<div class="box1"></div>
<div class="box2"></div>
<div class="box3"></div>
<script>
    // TweenMax.staggerTo([".box1", ".box2", ".box3"], 3,
 {x: 500}, 3);
    // TweenMax.staggerFrom([".box1", ".box2", ".box3"],
 3, {x: 500}, 3);
    TweenMax.staggerFromTo([".box1", ".box2", ".box3"],
3, {x: 500}, {x: 200}, 3);
</script>
</body>
</html>
```

–

41. GSAP动画属性

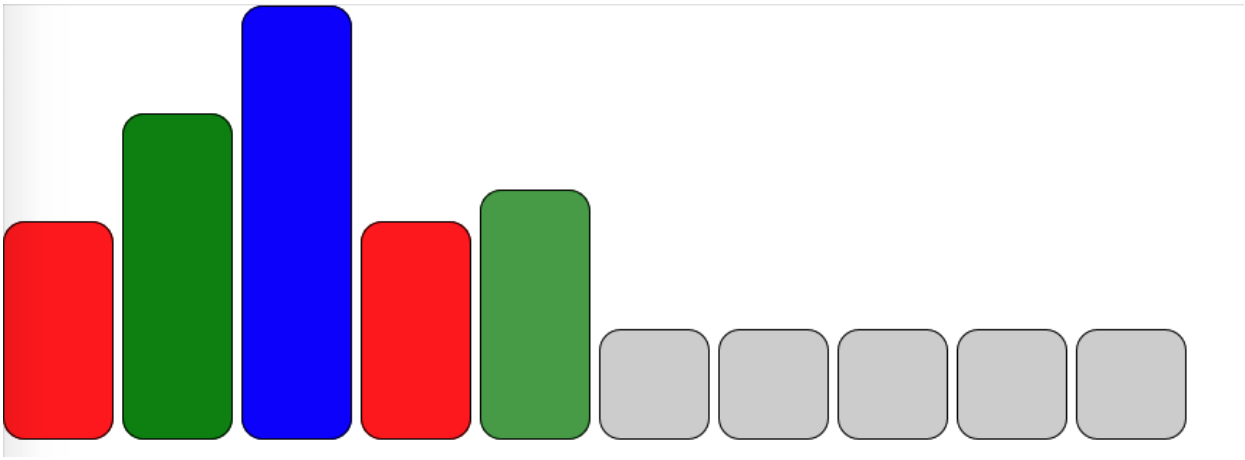
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>03-GSAP动画属性</title>
    <script src="js/TweenMax.js"></script>
    <style>
        *{
            margin: 0;
            padding: 0;
        }
        div{
            width: 100px;
            height: 100px;
            background: red;
        }
    </style>
</head>
<body>
<div class="box"></div>
<script>
    new TweenMax(".box", 3, {
        // 设置动画开始之前的延迟时间
        // delay: 2,
        // 设置动画初始值
        startAt: {
            x: 100
        },
        // 设置动画结束值
        css: {
            x: 500,
        },
        // 设置动画重复执行的次数
        // 无限重复 -1
        repeat: 2,
        // 设置动画重复执行的往返动画
        yoyo: true,
        // 设置重复动画开始之前的延迟时间
```

```
            repeatDelay: 3,
            // 设置动画执行的节奏
            ease: Bounce.easeOut,
            yoyoEase: Bounce.easeOut
        });
    </script>
    </body>
    </html>
```

42. GSAP循环动画

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>04-GSAP循环动画</title>
    <style>
        *{
            margin: 0;
            padding: 0;
        }
        div{
            display: inline-block;
            width: 50px;
            height: 50px;
            background: #ccc;
            border: 1px solid #000;
            border-radius: 10px;
        }
    </style>
    <script src="js/TweenMax.js"></script>
</head>
<body>
<div class="box"></div>
<div class="box"></div>
<div class="box"></div>
<div class="box"></div>
```

```
<div class="box"></div>
<div class="box"></div>
<div class="box"></div>
<div class="box"></div>
<div class="box"></div>
<div class="box"></div>
<script>
    let oDivs = document.querySelectorAll(".box");
    TweenMax.staggerTo(oDivs, 3, {
        cycle: {
            height: [100, 150, 200],
            backgroundColor: ["red", "green", "blue"]
        }
    }, 3);
</script>
</body>
</html>
```



43. GSAP动画事件

- 常用的有 `onStart`， `onComplete` 和 `onUpdate`。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>05-GSAP动画常用事件</title>
    <script src="js/TweenMax.js"></script>
```

```
    <style>
        *{
            margin: 0;
            padding: 0;
        }
        div{
            width: 100px;
            height: 100px;
            background: red;
        }
    </style>
</head>
<body>
<div class="box"></div>
<script>
    let obj = {name: "lnj"};
    TweenMax.to(".box", 3, {
        x: 500,
        delay: 3,
        onStart:function(a, b, c){
            // console.log("动画开始了", a, b, c);
            console.log(this);
        },
        onStartParams:["123", "456", "789"],
        onStartScope: obj,
    });
</script>
</body>
</html>
```

44. GSAP动画方法

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>06-GSAP动画常用方法</title>
```

```
    <script src="js/TweenMax.js"></script>
    <style>
        *{
            margin: 0;
            padding: 0;
        }
        div{
            width: 100px;
            height: 100px;
            background: red;
        }
    </style>
</head>
<body>
<div class="box"></div>
<button class="start">开始</button>
<button class="paused">暂停</button>
<button class="toggle">切换</button>
<button class="restart">重新开始</button>
<script>
    let tm = TweenMax.to(".box", 3, {
        x: 500,
        paused: true
    });
    // console.log(tm);
    let oStartBtn = document.querySelector(".start");
    oStartBtn.onclick = function () {
        tm.play();
    }

    let oPauseBtn = document.querySelector(".paused");
    oPauseBtn.onclick = function () {
        tm.pause();
    }

    let oToggleBtn = document.querySelector(".toggle");
    oToggleBtn.onclick = function () {
        // tm.paused(true);
        // tm.paused(false);
        // console.log(tm.paused());
        tm.paused(!tm.paused());
```

```
    }

    let oRestartBtn = document.querySelector(".restart");
    oRestartBtn.onclick = function () {
        tm.restart();
    }
</script>
</body>
</html>
```

45. GSAP动画管理

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>07-GSAP动画管理</title>
    <script src="js/TweenMax.js"></script>
    <!--<script src="js/TimelineMax.js"></script>-->
    <style>
        *{
            margin: 0;
            padding: 0;
        }
        div{
            width: 100px;
            height: 100px;
            background: #ccc;
            border: 1px solid #000;
        }
    </style>
</head>
<body>
<div class="box1"></div>
<div class="box2"></div>
<div class="box3"></div>
<script>
```

```
    /*
    TweenMax.staggerTo([".box1", ".box2", ".box3"], 3, {
        x: 500
    }, 3);
    */
    /*
    TweenMax.to(".box1", 4, {
        x: 500
    });
    TweenMax.to(".box2", 3, {
        x: 400,
        delay:4
    });
    TweenMax.to(".box3", 3, {
        x: 300,
        delay:7
    });
    */
    let tm = new TimelineMax();
    tm.add(
        TweenMax.to(".box1", 4, {
            x: 500
        })
    );
    tm.add(
        TweenMax.to(".box2", 3, {
            x: 400
        })
    );
    tm.add(
        TweenMax.to(".box3", 3, {
            x: 300
        })
    );
</script>
</body>
</html>
```

46. Velocity开篇

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>08-Velocity开篇</title>
    <style>
        *{
            margin: 0;
            padding: 0;
        }
        div{
            width: 100px;
            height: 100px;
            background: red;
        }
    </style>
    <script src="js/jquery-3.1.1.js"></script>
    <script src="js/velocity.js"></script>

</head>
<body>
<div class="box"></div>
<script>
    /*
    1.什么是Velocity?
    Velocity 是一个简单易用、性能极高、功能丰富的轻量级JS动画库。
    它能和 jQuery/Zepto 完美协作，并和$.animate()有相同的 API，  但它不依赖 jQuery，可单独使用。
    Velocity 不仅包含了 $.animate() 的全部功能，  还拥有：颜色动画、转换动画(transforms)、循环、 缓动、SVG 动画、和 滚动动画 等特色功能

    官方地址: https://github.com/julianshapiro/velocity
    中文文档: http://shouce.jb51.net/velocity/index.html

    2.GSAP基本使用
    1.1导入Velocity文件
    1.2利用Velocity实现动画
```

```
    * */
    // 1.单独使用
    /*
    let oDiv = document.querySelector(".box");
    Velocity(oDiv, {
        height: "300px"
    }, {
        duration: 3000
    });
    */

    // 2.配合jQuery使用
    // 注意点: 必须先导入jQuery, 再导入velocity
    $(".box").velocity({
        height: "300px"
    }, {
        duration: 3000
    });
</script>
</body>
</html>
```

47. Velocity配置

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>09-Velocity常用配置</title>
    <style>
        *{
            margin: 0;
            padding: 0;
        }
        div{
            width: 100px;
            height: 100px;
```

```
            background: red;
        }
    </style>
    <script src="js/jquery-3.1.1.js"></script>
    <script src="js/velocity.js"></script>
</head>
<body>
<div class="box"></div>
<script>
    $(".box").velocity({
        marginLeft: "500px"
    }, {
        duration: 3000,
        // 设置动画开始之前的延迟时间
        // delay: 2000,
        // 设置动画循环的次数
        // 注意点: 从初始位置到指定位置再到初始的位置算一次
        // loop: 2,
        // 设置动画运动的节奏
        // easing: "easeInOutQuint",
        // 设置动画结束之后元素的状态
        // display: "none",
        // visibility: "hidden"
        // 设置动画队列
        // 注意点: 只要设置了动画队列动画就不会自动执行
        queue: "a"
    });

    $(".box").velocity({
        marginTop: "500px"
    }, {
        duration: 3000,
        queue: "b"
    });
    $(".box").dequeue("a");
    setTimeout(function () {
        $(".box").dequeue("b");
    }, 3000)
</script>
</body>
</html>
```

- `display: none` 不会占用原有位置
- `visibility: hidden` 会占用原有位置

---

48. Velocity事件

- `begin`，`complete`，`progress` 是三个事件
- `elements` 是当前正在执行动画的元素，被放到了一个数组里面

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>10-Velocity常用事件</title>
    <style>
        *{
            margin: 0;
            padding: 0;
        }
        div{
            width: 100px;
            height: 100px;
            background: red;
        }
    </style>
    <script src="js/jquery-3.1.1.js"></script>
    <script src="js/velocity.js"></script>
</head>
<body>
<div class="box"></div>
<script>
    $(".box").velocity({
        marginLeft: "500px"
    }, {
        duration: 3000,
        delay: 2000,
        begin: function(elements) {
            console.log("动画开始了", elements);
        },
```

```
            complete: function(elements) {
                console.log("动画结束了", elements);
            },
            /*
            elements：当前执行动画的元素，可以用$(elements)来获取
            complete：整个动画过程执行到百分之多少，该值是递增的，注
意：该值为0-1的十进制小数 并不带单位 (%)
            remaining：整个动画过程还剩下多少毫秒，该值是递减的
            start：动画开始时的绝对时间 (Unix time)
            tweenValue：动画执行过程中 两个动画属性之间的补间值
            * */
            progress: function(elements, complete, remaining,
 start, tweenValue) {
                // console.log("动画正在执行");
                console.log((complete * 100) + "%");
            }
        });
    </script>
    </body>
    </html>
```

完成于 `2019.9.13`