链滴

javascript开发日历

作者：crick77

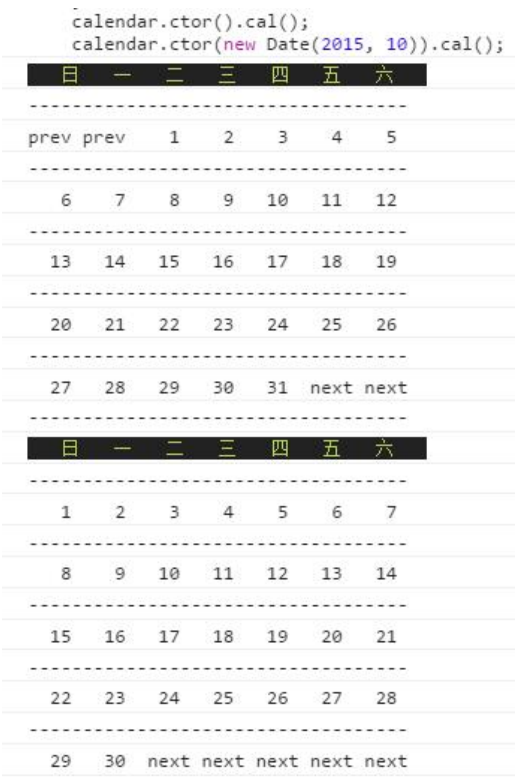
原文链接：https://ld246.com/article/1450444497475

来源网站：链滴

起因

需要实现一个日历功能，网上找了几个示例，都是根据各种库和插件，居然没有纯净的js完成的日历件，不免有些诧异，正好有时间，准备通过js编写一个简单的日历demo基础，可自由根据使用的插进行显示层的定制。

效果图



前提

算法

说一下日历的算法

1. (本月第一天的星期数+本月的天数)/7 可以知道本月需要占据几行。

2. 将日历看为二维数组，第一级遍历条件为本月行数，第二级为0~7（直观形象为一个日历table，左向右，自上而下循环）。

3. n为0~7循环参数， 计算当前日历方格显示数值为当前日期： 行数*7 + n - 本月第一天 + 1。

4. 二月的天数根据闰年会有不同，闰年的计算方式： 当前年份能被400整除，或者当前的年份能被4除且不能被100整除。

Date

日历功能实现必须通过js的Date对象提供的基础数值。介绍一下用到的方法功能

```
new Date().getFullYear(); //当前年份 2015
new Date().getMonth(); //当前月份 0~11
new Date().getDate(); //当前月中的某一天 1~31
new Date().getDay(); //一周中的某一天 0~6
new Date(year, month, day); //根据构造函数返回Date对象
```

实现

这里写出来编写代码思路分析的过程，可以跳过直接看下方的完整代码。

先抽象出二维数组模型，7列已经确定，需要先根据**算法1**算出本月的日历行数，但是二月的天数不一，所以先要根据是否为闰年算出二月的天数。

```
var calendar = {
    isLeap: function(vYear) {
        return vYear % 400 == 0 || (vYear % 4 == 0 && vYear % 100 != 0);
    },
    getFebruaryDays: function(vYear) {
        return this.isLeap(vYear) ? 29 : 28;
    }
}
```

本月第一天的星期数可以根据new Date(currentYear, currentMonth, 1).getDay()得出。

monthFirstDay为本月第一天的星期数，monthDays为12月的天数组成的数组。根据构造函数计算值，并返回this对象方便链式调用。

```
var calendar = {
    year: null,
    month: null,
    date: null,
    day: null,
    monthDays: null,
    monthFirstDay: null,
    ctor: function(vNow) {
        var now = vNow || new Date(); //如果为指定date对象，则为当前日期对象
        this.year = now.getFullYear();
        this.month = now.getMonth();
        this.date = now.getDate();
        this.day = now.getDay();
        this.monthFirstDay = new Date(this.year, this.month, 1).getDay();
        this.monthDays = [
            31, this.getFebruaryDays(this.year), 31, 30, 31, 30, 31, 31, 30, 31, 30, 31
        ];
        return this;
    },
    isLeap: function(vYear) {
        return vYear % 400 == 0 || (vYear % 4 == 0 && vYear % 100 != 0);
    },
    getFebruaryDays: function(vYear) {
        return this.isLeap(vYear) ? 29 : 28;
    }
}
```

计算日历行数

```
getMonthDay: function() {
    return this.monthDays[this.month];
},
getCalTr: function() {
    return (this.monthFirstDay + this.getMonthDay()) / 7;
},
```

循环二维数组得到日历, *console.log(n);*可结合使用的框架进行输出展示。

```
cal: function() {
    var that = this;
    for (var i = 0; i < this.getCalTr(); i++) {
        for (var j = 0; j < 7; j++) {
            var n = (i * 7 + j) - that.monthFirstDay + 1;

            if (n > 0 && n <= that.getMonthDay()) {
                console.log(n);
            }
        }
    }
}
```

完整代码，在控制台查看输出的日历

```
var calendar = {
    year: null,
    month: null,
    date: null,
    day: null,
    monthDays: null,
    monthFirstDay: null,
    days: [
        "星期一",
        "星期二",
        "星期三",
        "星期四",
        "星期五",
        "星期六",
        "星期日"
    ],

    ctor: function(vNow) {
        var now = vNow || new Date(); //如果为指定date对象，则为当前日期对象
        this.year = now.getFullYear();
        this.month = now.getMonth();
        this.date = now.getDate();
        this.day = now.getDay();
        this.monthFirstDay = new Date(this.year, this.month, 1).getDay();
        this.monthDays = [
```

```
            31, this.getFebruaryDays(this.year), 31, 30, 31, 30, 31, 31, 30, 31, 30, 31
        ];
        return this;
    },
    isLeap: function(vYear) {
        return vYear % 400 == 0 || (vYear % 4 == 0 && vYear % 100 != 0);
    },
    getFebruaryDays: function(vYear) {
        return this.isLeap(vYear) ? 29 : 28;
    },
    format: function() {
        return (this.month + 1) + "/" + this.date + "/" + this.year + "  " + this.dayMap[this.day];
    },
    getDay: function() {
        return this.days[this.day - 1]
    },
    getMonthDay: function() {
        return this.monthDays[this.month];
    },
    getCalTr: function() {
        return (this.monthFirstDay + this.getMonthDay()) / 7;
    },
    cal: function() {
        var that = this;
        console.log("%c  日  一  二  三  四  五  六  ",'background: #222; color: #bada55');
        console.log("-----------------------------------");
        for (var i = 0; i < this.getCalTr(); i++) {
            var temp = "";
            for (var j = 0; j < 7; j++) {
                var tempData = (i * 7 + j) - that.monthFirstDay + 1
                if (tempData < 1) {
                    temp = temp + "prev ";
                } else if (tempData > that.getMonthDay()) {
                    temp = temp + " next";
                } else {
                    if (tempData > 9) {
                        temp = temp + "  " + tempData + " ";
                    } else {
                        temp = temp + "   " + tempData + " ";
                    }
                }
            }
            console.log(temp);
            console.log("-----------------------------------");
        }
    }
}
calendar.ctor().cal();
calendar.ctor(new Date(2015, 10)).cal();
```

总结

不足

本例中的ctor并非构造函数，而是针对一个对象进行变更，重新执行ctor后会替换calendar的属性值。

未对之前和之后的日期进行获取，如需相关功能，可在ctor中设置prevMonth 和 nextMonth的值，要注意跨年度问题，时间和需求问题，暂未实现。

思考

是否采用构造函数的方式生成多个对象？ （如果是工具类的话，不需要构造函数）

方法中采用this.param 还是获取外部参数 如何界定?

收获

用面向对象的方式编写js，思路清晰，语感更好。