

## getMonth ( ) &setMonth ( )

	getMonth ( )	SetMonth ( )
<b>定义</b>	返回表示月份的数字	用于设置月份
<b>用法</b>	dateObj.getMonth ( )	dateObj.getMonth ( month , day )
<b>参数</b>	无	Month : 必需要有。是一个表示月份的数值 , 这个数值是 0 ( 一月 ) ~11 ( 十二月 ) 之间  -1 为去年的最后一个月  12 为明年的第一个月  13 为明年的第二个月
		day : 可选。表示月份的某一天的数值 , 这个数值是 1~31 之间  如果当月有 31 天 , 32 为下个月的第一天  如果当月有 30 天 , 32 为下个月的第二天
<b>返回值</b>	0 ( 一月 ) ~11 ( 十二月 ) 之间的一个整数	调整过的日期的毫秒数

### ( 1 ) getMonth ( )

例如 : var today = new Date ( );

Console.log(today . getMonth ( ) );

得到的是比月份小 1 的数值 , 比如现在是 2016 年的 6 月 , 那么得到的数值就是 5

---

( 2 ) setMonth ( )

**例 1 : 只有参数 month**

```
var today = new Date ( );  
  
console.log ( today . setMonth ( 2 );  
  
console.log ( today . getMonth ( ));
```

打印结果 :

1457401927292

2

>

分析 :

today . setMonth ( 2 ) 设置的是当前的月份时间距离 1970 年 1 月 1 日零点的毫秒数 ( 毫秒数是随着你打印时间的变化而变化的 ); 将月份的数值设置为 2 , 那么 today . getMonth ( ) 的值也为 2 , 实际月份是 3 月 ( 因为 getMonth ( ) 的数值是 0~11 , 对应的是 1~12 月 , 所以数值 2 对应的是 3 月 )

**例 2 : 参数 month 和 day**

```
var today = new Date ( );  
  
today . setMonth ( 3,32 );  
  
console.log ( today . getMonth ( ));  
  
console.log ( today . getDate ( ));
```

打印结果 :

4

2

>

### 分析：

- 本例中，如果只设置 1 个月份参数：setMonth ( 3 )  
getMonth ( ) 的值就是 3 , 实际月份是 4 月 ( 因为 getMonth ( ) 的数值是 0~11 , 对应的是 1~12 月 , 所以数值 3 对应的是 4 月 );
- 可是本例中设置了月份和天数两个参数：setMonth ( 3,32 ) ,  
按照上面的分析 , 目前的月份是 4 月 , 4 月份的天数是 30 天 ;  
但是第二个参数值超过了 30 , 32-30 余 2 , 所以月份进 1 , 天数剩 2。
- 那么目前的月份就是 5 月 , 日期就是 2

### 总结：

针对 setMonth ( ) 这个方法，我们整理了如下公式：

注：当前年份为 2016

参数只有月份：setMonth ( month )

方法	参数	浏览器显示	实际月份	年份
setMonth(参数)	12 为基数	参数/12 , 取 余数	( 数/12 , 取余 数 ) +1	( 数/12 , 取整 ) + 当 前年份
例子：	6	6	7	2016
	13	1	2	2017

setMonth(参数)	28	4	5	2018
	32	8	9	2019

参数有两个：setMonth ( month , day )

	月 份 31 天 , 1, 3,5,7,8,10,12 这 几 个 月是31天	例 : setMonth(2,35)	月 份 30 天 , 4,6,8,9,11 这 几 个 月 是 30 天	例 : setMonth(3,35)	2 月 , 28 天	例 : setMonth(1,35) 年 份 设 置 为 : 1997	2 月 ( 闰 年 29 天 )	例 : setMonth(1,35) 年 份 设 置 为 : 2000
浏览器显示 month	当参数day<=31, 显示为month	3	当参数day<=30, 显示为month	4	当参数day<=28, 显示为month	2	当参数day<=29, 显示为month	4
	当参数day>31, 显示为 : month+1		当参数day>30, 显示为 : month+1		当参数day>28 , 显示为 : month+1		当参数day>29 , 显示为 : month+1	
实际月份	当参数day<=30, 实际月份month+1	4月	当参数day<=30, 实际月份month+1	5月	当参数day<=28, 实际月份month+1	3月	当参数day<=29, 实际月份 month+1	2月
	当参数day>30, 实际月份month+2		当参数day>30, 实际月份month+2		当参数day>28, 实际月份month+2		当参数day>29, 实际月份 month+2	
天数	当参数day<=31 天数为 : day	4	当参数day<=30 天数为 : day	5	当参数day<=28 天数为 : day	7	当参数day<=29 天数为 : day	6
	当参数day>31 天数=day-31		当参数day>30 天数=day-30		当参数day>28 天数=day-28		当参数day>29 天数=day-29	

补充：

上面的天数针对的是 day 的数值小于两个月的天数，当 day 的数值大于 2 个月的天数时，我们看下如何计算：

比如：var today=new Date();

today.setMonth(2,80);

console.log(today.getMonth());

console.log(today.getDate());

运行结果：

4
19
>

### 分析：

第一步：设置的 month 是 2，代表的是 3 月，3 月有 31 天

设置的 day 是 80，故月份先+1， $\text{day}-31=49$

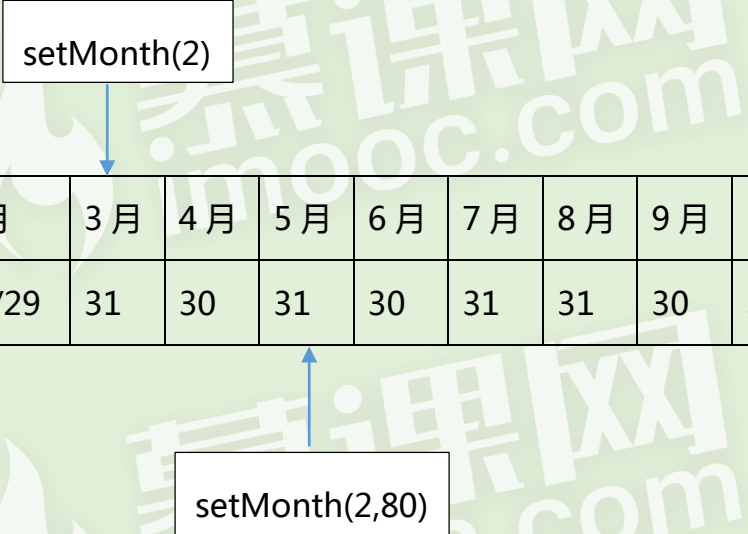
现在 month 是 3，day 是 49

第二步：month 是 3，代表是 4 月，4 月有 30 天

day 目前是 49，故月份再+1，天数剩  $\text{day}-30=19$ ；

到现在为止，天数已经减不下去了，我们得到的 month 是 4，实际月份是 5 月，天数是 19

### 图解：



月份	1 月	2 月	3 月	4 月	5 月	6 月	7 月	8 月	9 月	10 月	11 月	12 月
天数	31	28/29	31	30	31	30	31	31	30	31	30	31