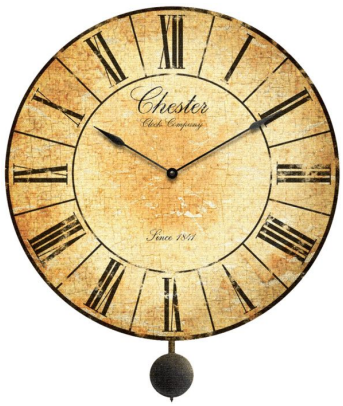
Date



> ! 本文介绍JavaScript中的内置对象Date，时间相关的基本常识，以及Date的常用方法，此外还简单介绍了定时器的相关知识点。

日期类型Date简单介绍

Date对象是JavaScript语言中内置的数据类型，日期实例对象通常都通过Date构造函数来创建。

[Date对象]相关的方法非常多，但总体来说这些方法大概可以分成两类，即对(日期)时间进行设置和读取操作，具体的操作则会涉及到时间的 年-月-日 时-分-秒-毫秒以及星期 等维度。因为在编码中操作的时间可能是本地时间或UTC世界时间(GMT)，因此Date对象几乎提供了两套相同用处的方法(方法名中有UTC的表示操作的是世界时间，没有UTC的表示操作的是本地时间)。

本文会先简单列出Date对象的相关方法，然后再具体讲解日期对象的创建、开发中常见的处理场景以及核心方法的使用。

```
console.dir(Date)
ƒ Date()
    UTC: ƒ UTC()                    /*将Date转换为毫秒格式 参照1970-01-01午夜(GMT)*/
    arguments: (...)
    caller: (...)
    length: 7                       /*Date构造函数的形参个数*/
    name: "Date"                    /*Date构造函数的名字*/
    now: ƒ now()                    /*以毫秒形式返回当前时间*/
    parse: ƒ parse()                /*解析一个日期/时间字符串*/
    prototype:                      /*Date构造函数原型对象的成员*/
        constructor: ƒ Date()
        getDate: ƒ getDate()                        /*返回日期值*/
        getDay: ƒ getDay()                          /*返回星期值*/
        getFullYear: ƒ getFullYear()                /*返回年份值*/
        getHours: ƒ getHours()                      /*返回小时值*/
        getMilliseconds: ƒ getMilliseconds()        /*返回毫秒值*/
        getMinutes: ƒ getMinutes()                  /*返回分钟值*/
        getMonth: ƒ getMonth()                      /*返回月份值*/
        getSeconds: ƒ getSeconds()                  /*返回秒钟值*/
```

```
    getTime: ƒ getTime()                    /*返回时间戳(毫秒形式的时间)*/
    getTimezoneOffset: ƒ getTimezoneOffset()  /*返回与GMT的时间差*/
    getUTCDate: ƒ getUTCDate()              /*返回日期值(全球时间)*/
    getUTCDay: ƒ getUTCDay()                /*返回星期值(全球时间)*/
    getUTCFullYear: ƒ getUTCFullYear()      /*返回年份值(全球时间)*/
    getUTCHours: ƒ getUTCHours()            /*返回小时值(全球时间)*/
    getUTCMilliseconds: ƒ getUTCMilliseconds()/*返回毫秒值(全球时间)*/
    getUTCMinutes: ƒ getUTCMinutes()        /*返回分钟值(全球时间)*/
    getUTCMonth: ƒ getUTCMonth()            /*返回月份值(全球时间)*/
    getUTCSeconds: ƒ getUTCSeconds()        /*返回秒钟值(全球时间)*/
    getYear: ƒ getYear()                    /*返回年份值 = 当前年份-1900*/
    setDate: ƒ setDate()                    /*设置日期*/
    setFullYear: ƒ setFullYear()            /*设置年份、可选的月份和日期*/
    setHours: ƒ setHours()                  /*设置时间的小时，可选的分钟、秒以及毫秒*/
    setMilliseconds: ƒ setMilliseconds()    /*设置时间的毫秒值*/
    setMinutes: ƒ setMinutes()              /*设置时间的分钟、可选的秒钟以及毫秒*/
    setMonth: ƒ setMonth()                  /*设置时间的月份和日期*/
    setSeconds: ƒ setSeconds()              /*设置时间的秒钟和可选的毫秒*/
    setTime: ƒ setTime()                    /*使用毫秒值来设置时间*/
    setUTCDate: ƒ setUTCDate()              /*设置日期值(全球时间)*/
    setUTCFullYear: ƒ setUTCFullYear()      /*设置年，可选的月份和日期(全球时间)*/
    setUTCHours: ƒ setUTCHours()            /*设置小时，可选的分钟、秒钟和毫秒(全球时间)*/
    setUTCMilliseconds: ƒ setUTCMilliseconds()/*设置毫秒值(全球时间)*/
    setUTCMinutes: ƒ setUTCMinutes()        /*设置分钟、可选的秒钟和毫秒(全球时间)*/

    setUTCMonth: ƒ setUTCMonth()            /*设置月份和日期(全球时间)*/
    setUTCSeconds: ƒ setUTCSeconds()        /*设置秒钟和毫秒(全球时间)*/
    setYear: ƒ setYear()                    /*设置年份*/
    toDateString: ƒ toDateString()          /*以字符串的形式返回日期*/
    toGMTString: ƒ toUTCString()            /*转换为全球时间表示的字符串*/
    toISOString: ƒ toISOString()            /*转换为ISO-8601格式的字符串*/
    toJSON: ƒ toJSON()                      /*JSON序列化日期对象*/
    toLocaleDateString: ƒ toLocaleDateString()/*返回本地格式的日期字符*/
    toLocaleString: ƒ toLocaleString()      /*转换为本地格式的字符串*/
    toLocaleTimeString: ƒ toLocaleTimeString()/*转换为本地格式表示的Date时间部分*/
    toString: ƒ toString()                  /*转换为字符串*/
    toTimeString: ƒ toTimeString()          /*以字符串形式返回date的时间部分*/
    toUTCString: ƒ toUTCString()            /*转换为字符串(全球时间)*/
    valueOf: ƒ valueOf()                    /*获取日期的时间戳 等价于getTime()*/
    Symbol(Symbol.toPrimitive): ƒ [Symbol.toPrimitive]()
    __proto__: Object
__proto__: ƒ ()
[[Scopes]]: Scopes[0]
```

日期类型Date的基本使用

创建Date实例对象

通常我们使用构造函数Date来获取日期(时间)实例对象，在调用构造函数的时候有多种传参方式。

`new Date()` 根据当前时间来创建Date实例。

`new Date(milliseconds)` 根据 时间戳 (毫秒计数的纯数字)来创建Date实例。

`new Date(dateString)` 根据特殊格式的日期字符串也能创建Date实例。

`new Date(year,month,[day],[hours],[minutes],[seconds],[ms])` 根据年月日时分秒来创建实例。

```
/*日期Date实例的创建的N种方式*/
/*01-获取当前时间*/
var time1 = new Date();

/*02-根据时间戳来创建Date对象*/
var milliseconds = time1.getTime();
var time2 = new Date(milliseconds);

/*03-传入7个参数(年月日时分秒和毫秒创建)*/
var time3 = new Date(2019,5,20,1,3,1,4);

/*04-根据固定格式的日期字符串来创建*/
var dateString1 = "2019-09-01";
var dateString2 = "2019/09/02";
var dateString3 = "October 13, 1975 11:13:00";
var time4 = new Date(dateString1);
var time5 = new Date(dateString2);
var time6 = new Date(dateString3)

/*05-Date在调用的时候可以缺省new关键字*/
var time7 = Date();     //获取的是当前的时间

/*日期对象中包含的信息主要有：年、月、日、时、分、秒、毫秒、星期和时区*/
console.log(time1);     //Wed Apr 17 2019 10:54:24 GMT+0800 (中国标准时间)
console.log(time2);     //Wed Apr 17 2019 10:54:24 GMT+0800 (中国标准时间)
console.log(time3);     //Thu Jun 20 2019 01:03:01 GMT+0800 (中国标准时间)

console.log(time4);     //Sun Sep 01 2019 08:00:00 GMT+0800 (中国标准时间)
console.log(time5);     //Mon Sep 02 2019 00:00:00 GMT+0800 (中国标准时间)
console.log(time6);     //Mon Oct 13 1975 11:13:00 GMT+0800 (中国标准时间)
console.log(time7);     //Wed Apr 17 2019 11:04:59 GMT+0800 (中国标准时间)

/*把本地时间转换为GMT全球时间格式打印*/
time5.toUTCString();    //"Sun, 01 Sep 2019 16:00:00 GMT"
```

获取并格式化时间

在开发中我们经常需要以特定的格式来表示时间，譬如 `2008-08-08 08: 08: 08` 或者是 `2008/08/08` 。其实参考上文的代码，我们发现Date对象为我们提供了一大堆方法来获取Date实例中具体的年月日时分秒等数值，通过调用这些方法我们可以把日期格式化成任何我们想要的形式。

```
/*01-获取当前的Date时间对象*/
var date  = new Date();

/*02-获取具体的每个值*/
var year  = date.getFullYear();       /*年*/
var month = date.getMonth()+1;        /*月*/
var day   = date.getDate();           /*日*/
var hour  = date.getHours();          /*时*/
var min   = date.getMinutes();        /*分*/
var sec   = date.getSeconds();        /*秒*/

/*03-按照固定的格式拼接*/
var res = year + '-' + month + '-' + day + ' ' + hour + ':' + min + ':' + sec;
```

```
/*2019-4-17 11:31:18*/
console.log(res);
```

在上面的代码中，我们把当前的日期对象转换成了固定格式的字符串 `2019-4-17 11:31:18` ，其实除了像上面那样拼接外我们还可以简单的通过数组的 `join()` 方法来执行格式化操作。此外，有时候我们可能需要保证年月日时分秒等占两位，即上面的时间应该表示为 `2019-04-17 11:31:18` ，那么可能还需要对某些数据执行 补零操作。

```
/*00-封装补零的方法*/
function prefixInteger(num) {
  return (Array(2).join('0') + num).slice(-2);
}

/*01-获取当前的Date时间对象*/
var date  = new Date();

/*02-获取具体的每个值*/
var year  = date.getFullYear();       /*年*/
var month = prefixInteger(date.getMonth()+1);        /*月*/
var day   = prefixInteger(date.getDate());           /*日*/
var hour  = prefixInteger(date.getHours());          /*时*/
var min   = prefixInteger(date.getMinutes());        /*分*/
var sec   = prefixInteger(date.getSeconds());        /*秒*/
var week  = date.getDay();                            /*星期 0~6*/
var weStr = '日一二三四五六';

/*03-按照固定的格式拼接*/
var arr = [];
arr.push(year + "-");
arr.push(month + "-");
arr.push(day + " ");
arr.push(hour + ":");
arr.push(min + ":");
arr.push(sec + " ");
arr.push("星期"+ weStr[week]);

var res = arr.join("")
console.log(res);       /*2019-04-17 11:56:12 星期三*/
```

Date对象的常用方法

`toUTCString()` 方法把时间转换成UTC时间。

`Date.now()` 方法返回执行这行代码时距 1970-01-01 0点的毫秒数。

`Date.parse()` 返回指定日期距1970-01-01 0点的毫秒数，默认支持 `年-月-日` 或 `年/月/日` 格式。

`toLocaleDateString()` 方法以特定的地区格式显示年、月、日。

`getTime()` | `setTime()` 方法用来获取 | 修改某个日期自1970年1月1日0时以来的毫秒数(时间戳)。

```
/*获取当前时间*/
var date = new Date();
```

```
/*傻傻分不清楚的时间字符串*/
console.log(date);                      //Wed Apr 17 2019 15:19:03 GMT+0800 (中国标准时间)
console.log(date.toDateString());       //Wed Apr 17 2019
console.log(date.toISOString());        //2019-04-17T07:19:03.303Z
console.log(date.toLocaleTimeString()); //下午3:19:03
console.log(date.toLocaleDateString()); //2019/4/17


/*获取时间戳*/
console.log(date.getTime());            //1555485543303

console.log(Date.parse("2008-08-08"));  //1218153600000
console.log(Date.parse("2008/08/08"));  //1218124800000

/*把时间戳转换为时间字符串格式*/
var resDate = new Date(Date.parse("2008/08/08"));
console.log(resDate);                   //Fri Aug 08 2008 00:00:00 GMT+0800 (中国标准时间)
```

示例 给定两个日期，计算它们之间间隔的天数。

```
/*01-提供两个时间(注意格式)*/
var start = "2019-04-01";
var end   = "2019-04-22"

/*02-计算两个时间见的差值 毫秒数据*/
var offset = Date.parse(end) - Date.parse(start);

/*03-把差值转换为天数*/
var days = Math.ceil(offset/1000/60/60/24);

/*04-拼接并输出*/
var res = start +' 至 '+ end + ' 还差: ' + days + "天";
console.log(res);   //2019-04-01 至 2019-04-22 还差: 21天
```

定时器和延迟执行

JavaScript中提供了专门的定时器和延迟执行的方法，我们简单介绍下它们的基本语法。

延迟执行函数

语法 `setTimeout(code,millisec)`

作用 `setTimeout()` 方法用于在指定的毫秒数后调用函数或计算表达式

说明 `setTimeout()` 方法只会执行 code 一次。

取消 当延迟执行函数启动后可以通过调用对应的 `clearTimeOut()` 方法传入参数来关闭定时器。

```
/*01-延迟执行函数的基本使用*/
 console.log("start",new Date());
 setTimeout(function () {
   /*1秒钟后才执行回调函数中的代码*/
   console.log("task",new Date());
 },1000)

 console.log("end",new Date());
```

```
console.log('end',new Date());

//start Wed Apr 17 2019 14:49:38 GMT+0800 (中国标准时间)
//end   Wed Apr 17 2019 14:49:38 GMT+0800 (中国标准时间)
//task  Wed Apr 17 2019 14:49:39 GMT+0800 (中国标准时间)
```

示例 演示如果使用 `setTimeout()` 延迟执行函数来实现时钟的功能。

HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
</head>
<body onload="startTime()">
<div id="demo"></div>
<script>

  function startTime()
  {
    /*01-获取当前的时间*/
    var date = new Date()

    /*02-获取时分秒等信息并格式化*/
    var h    = checkTime(date.getHours());
    var m    = checkTime(date.getMinutes());
    var s    = checkTime(date.getSeconds());

    /*03-输出到页面中显示出来*/
    document.getElementById('demo').innerHTML=h+":"+m+":"+s

    /*04-延迟执行 设置500毫秒后重新调用startTime方法 */
    setTimeout('startTime()',500)
  }

  /*用来保证时分秒占2位，不足则补零 */
  function checkTime(i)
  {
    if (i<10) i="0" + i
    return i
  }
</script>
</body>
</html>
```

定时器函数

语法 `setInterval(code,millisec[,"lang"])`

作用 `setInterval()` 方法会按照指定的周期（以毫秒计）来调用函数或计算表达式。

说明 `setInterval()` 方法会不停地调用函数，直到 clearInterval() 被调用或窗口被关闭。

取消 当定时器开启后可以通过调用对应的 `clearInterval()` 方法传入参数来关闭定时器。

场景 定时器函数常被用来处理一些需要长时间执行的重复性工作，譬如轮播图。

```
/*开启定时器*/
```

```
var timer = setInterval(function () {
  console.log("--滴答--滴答--! ");
},1000);

/*取消定时器*/
// clearInterval(timer);
// 说明 上面的代码执行后每隔1秒就打印一次 "--滴答--滴答--! "
```

附录：时间有关的常识

纪元 纪元时间(UNIX TIME)指的是 1970-01-01 00:00:00

GMT 格林尼治标准时间 (Greenwich Mean Time)，俗称“天文学时间”

它根据地球的自转和公转来计算时间，也就是太阳每天经过位于英国伦敦郊区的皇家格林尼治天文台的时间就是中午12点，而地球的自转正在缓速变慢，所以时间误差比较大。

UTC 世界协调时间 (Universal Time Coordinated)，又称为“原子物理时间”。

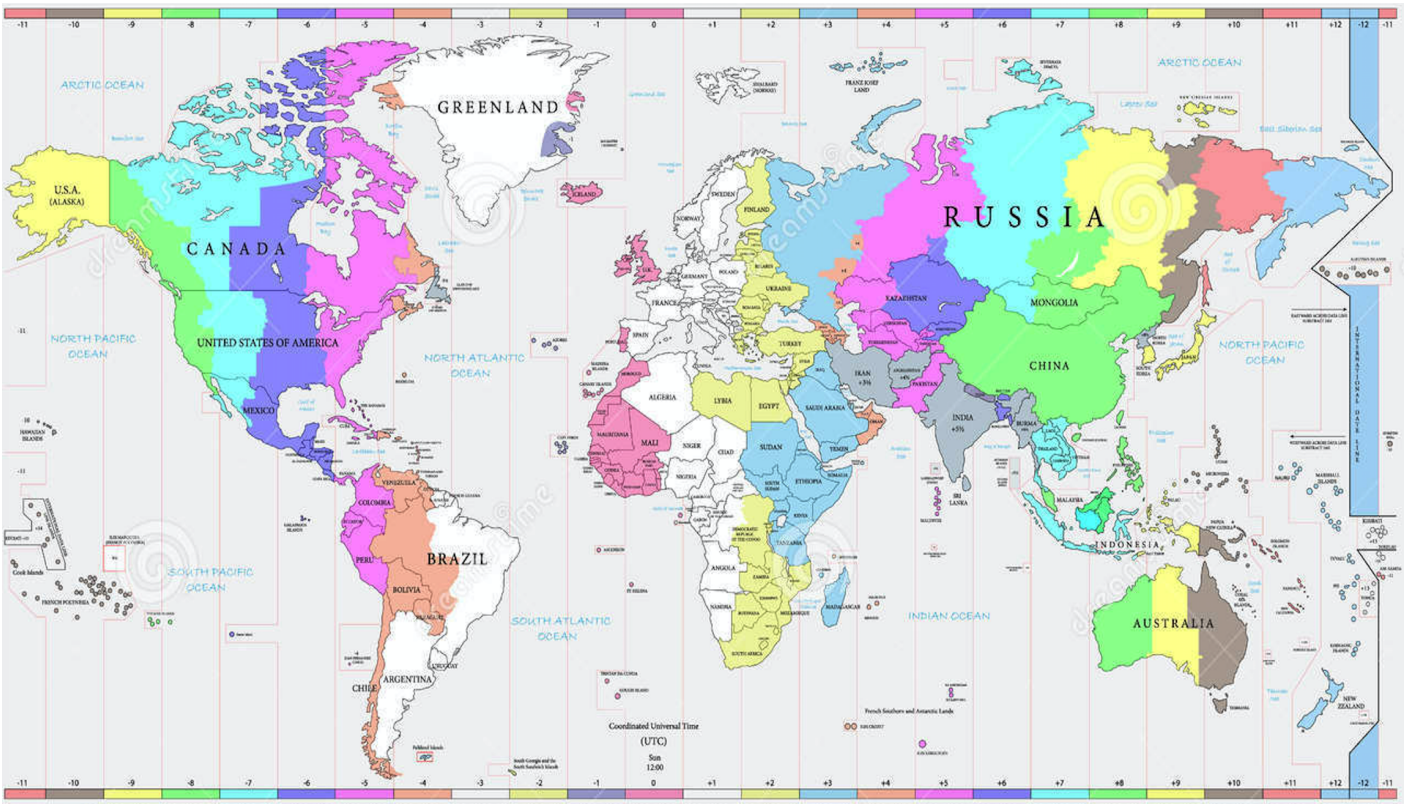
世界协调时间更加精确，50亿年才误差1秒。

闰年 四年一闰，百年不闰，四百年再闰

闰年的作用是为了修正计时的偏差，地球绕太阳一周的准确时间是365天5小时48分46秒，用小数表示即365.24219天，为了方便我们平时计算一年使用的是365天，这样实际每年就会多出0.24219天，四年就多了一天，如果按四年年一润的算法，100年总共有25个闰年，但实际100年只多出24.219天，所以就有了百年不润的说法；因此，我们在判断闰年的法则为 ① 能被4整除且不能被100整除的为闰年 ② 能被400整除的是闰年。

时区 全球共被划分为24个时区，我们使用的是北京时间(东八区)

为了克服时间上的混乱，1884年在华盛顿召开的一次国际经度会议（ 国际子午线会议 ）上，规定将全球划分为24个时区（东西各12个时区）。规定英国（格林尼治天文台旧址）为中时区（零时区）、东1-12区，西1-12区。每个时区横跨经度15度，时间正好是1小时



- Posted by 博客园·文顶顶 | 花田半亩
- 联系作者 简书·文顶顶 新浪微博·Coder_文顶顶
- 原创文章，版权声明：自由转载-非商用-非衍生-保持署名 | 文顶顶