ECMAScript 中的 Date 类型是在早期 Java 中的 java.util.Date 类基础上构建的。为此， Date
类型使用自 UTC（Coordinated Universal Time，国际协调时间）1970 年 1 月 1 日午夜（零时）开始经过的毫秒数来保存日期。在使用这种数据存储格式的条件下， Date 类型保存的日期能够精确到 1970 年 1月 1 日之前或之后的 285 616 年。

要创建一个日期对象，使用 new 操作符和 Date 构造函数即可，如下所示。

```
var now = new Date();
```

在调用 Date 构造函数而不传递参数的情况下，新创建的对象自动获得当前日期和时间。如果想根据特定的日期和时间创建日期对象，必须传入表示该日期的毫秒数（即从 UTC 时间 1970 年 1 月 1 日午夜起至该日期止经过的毫秒数）。为了简化这一计算过程，ECMAScript 提供了两个方法： Date.parse()和 Date.UTC() 。

Date.parse()

- “月/日/年”，如 6/13/2004；
- “英文月名 日,年”，如 January 12,2004；
- “英文星期几 英文月名 日 年 时:分:秒 时区”，如 Tue May 25 2004 00:00:00 GMT-0700。
- ISO 8601 扩展格式 YYYY-MM-DDTHH:mm:ss.sssZ（例如 2004-05-25T00:00:00）。只有兼容 ECMAScript 5 的实现支持这种格式。

例如，要为 2004 年 5 月 25 日创建一个日期对象，可以使用下面的代码：

```
var someDate = new Date(Date.parse("May 25, 2004"));
```

```
var someDate = new Date("May 25, 2004");
```

日期对象及其在不同浏览器中的实现有许多奇怪的行为。其中有一种倾向是将超出范围的值替换成当前的值，以便生成输出。例如，在解析"January 32, 2007"时，有的浏览器会将其解释为"February 1, 2007"。而 Opera 则倾向于插入当前月份的当前日期，返回"January 当前日期，2007"。也就是说，如果在 2007 年 9 月 21 日运行前面的代码，将会得到"January 21, 2007"（都是 21 日）。

Date.UTC()

Date.UTC() 的参数分别是年份、基于 0 的月份（一月是 0，二月是 1，以此类推）、月中的哪一天（1 到 31）、小时数（0 到 23）、分钟、秒以及毫秒数。在这些参数中，只有前两个参数（年和月）是必需的。如果没有提供月中的天数，则假设天数为 1；如果省略其他参数，则统统假设为 0。以下是两个使用 Date.UTC() 方法的例子：

```
// GMT 时间 2000 年 1 月 1 日午夜零时
var y2k = new Date(Date.UTC(2000, 0));

// GMT 时间 2005 年 5 月 5 日下午 5:55:55
var allFives = new Date(Date.UTC(2005, 4, 5, 17, 55, 55));
```

ECMAScript 5 添加了 Data.now()方法，返回表示调用这个方法时的日期和时间的毫秒数。这个方法简化了使用 Data 对象分析代码的工作。例如：

```
//取得开始时间
var start = Date.now();

//调用函数
doSomething();

//取得停止时间
var stop = Date.now(),
    result = stop - start;
```

var start = Date.now();