[#](https://github.com/mbostock/d3/wiki/Time-Formatting" \l "wiki-format) d3.time.**format**(*specifier*) 时间格式转换器（区分符）

以指定的区分符（specifier）构造一个时间格式器。

因为数据源里的数据都是字符串，比如“1-May-2012“，“Sunday,30,03,2014“，“Sunday,30,Ｍarch,2014“，要将其解析为Javascript的Date类型，或者将Date类型的时间转换为字符串，就需要定义一下format的时间格式转换器。

specifier，应该严格按照数据源里的时间字段来定义，

比如:

“Sunday,30,03,2014“，specifier为-----“%A,%d,%m,%Y“，

“Sunday,30,March,2014“，specifier为-----“%A,%d,%B,%Y“

var time = "Sunday,30,March,2014"; //这里的time相当于数据源，d.value

var formatter = d3.time.format("%A,%d,%B,%Y").parse

console.log(formatter(time)) ；

打印出来的结果是Sun Mar 30 2014 00:00:00 GMT+0800 (CST)

区分符应该用以下命令格式：

* %a - 简写的星期，Mon, Wen, Sat, Sun…
* %A - 全名的星期，Mondy,Sunday,Saturday…
* %b - 简写的月份，Mar,Apr,Jun…
* %B - 全名的月份，March, April, May…
* %c -日期和时间，等同于"%a %b %e %H:%M:%S %Y".
* %d - 包含0的日期，[01,31]
* %e -[ 1,31]; equivalent to %\_d.未包含0的日期，等同于%\_d; %d和%e其实没有区别，都能等同解析
* %H - 24小时格式的小时格式
* %I - 12小时格式的小时格式
* %j - [001,366].一年中的第几日[001,365]

"365,Sunday,1,March,2014"---"%j,%A,%d,%B,%Y",转换过来的时间是

Wed Dec 31 2014 00:00:00 GMT+0800 (CST) 也就是2014年最后一天，所以这里无论里的数据源里有没有月和日等详细的时间，只要你给定了一年中的第几天，结果只会指定到这一天

所以上面等同于"365，2014"---"%j,%Y",

* %m -[01,12].十进制的月份
* %M - 十进制的分
* %L - [000, 999].十进制的毫秒
* %p - AM or PM.
* %S - [00,61]. 十进制的秒
* %U - [00,53].十进制的周数，（周日为一周的第一天）
* %w - 十进制的星期，0为周日［0,6]
* %W - [00,53].十进制的周数，（周一为一周的第一天）
* %x - %x等同于’%m/%d/%Y’， ‘03/29/2014’ 注意这里的区分符里有个/
* %X - %X等同于"%H:%M:%S”
* %y - [00,99]，没有千年的年份，比如14即为2014
* %Y - 有千年的年份，如1999,2014
* %Z - 时区，比如："-0700".
* %% - %字符

对于%U，早于第一个周日的所有新年日期都视为第0周。比例2014年的第一个周日是1月5号，那之前的4天，1号至4号都为第0周

对于%W，早于第一个周一的所有新年日期都视为第0周。比例2014年的第一个周一是1月6号，那之前的5天，1号至5号都为第0周

在Python中，strftime和strptime的一些执行中，可能需要包括可选的宽度或精确度为命令；这一特性在Ｄ3中没有执行，但在将来可能会被增加进来。需要提示的是：Javascript环境没有提供标准的API给获取地区设定特性的日期和时间转换器，所以D3的执行方法是依据$LOCALE环境变量来解析时间的

%标识表明命令后面可能直接有一个填充修改器：

* 0 - 0填充
* \_ - 空格填充
* - - 取消填充

如果没有指定填充修改器，则所有的命令为0填充，除了%e默认为空格填充；

Format的返回值是一个对象，也是一个函数。例如：

**var** format **=** d3.time.format("%Y-%m-%d"); format.parse("2011-01-01"); *// returns a Date* format(**new** Date(2011, 0, 1)); *// returns a string*

[#](https://github.com/mbostock/d3/wiki/Time-Formatting" \l "wiki-_format) **format**(*date*)

将指定的日期格式化，返回一个相应的字符。这里的date参数必须是Javascript的Date对象。 注意，如果日期是用定量比例尺（ [quantitative scales](https://github.com/mbostock/d3/wiki/Quantitative-Scales)）相连接的，那么日期将被隐性强制转为自1970/1/1以后的毫秒数字。可以用如下方法，将数字与时间进行相互转换。

time **=** **+**date; *// convert a Date object to time in milliseconds* date **=** **new** Date(time); *// convert a time in milliseconds to a Date object*

如果你喜欢明确转换日期，你可以用JS中Date对象的getTime方法，但是用+操作符更加简捷，运行也更快捷。

[#](https://github.com/mbostock/d3/wiki/Time-Formatting" \l "wiki-parse) format.**parse**(*string*)

The %d and %e format specifiers are considered equivalent for parsing.

将指定的字符解析为相应的Javascript Date对象。如果解析失败，返回null。不同于“自然语言”中的时间解析器（包括Javascript的内核解析器），format.parse方法更加严格：如果string没有完全和相应的format specifier相匹配，该方法将返回null。例如：format是完整的 ISO 8601字符串“%Y-%m-%dT%H:%M:%SZ”，那么“2011-07-01T19:15:28Z”的时间字符串就能被正确解析为 Javascript的Date对象，而其它的如：but "2011-07-01T19:15:28", "2011-07-01 19:15:28" 和 "2011-07-01"，将返回null，尽管它们都是有效的ISO 8601标准时间。（注意最后的大写“Z”不同于%Z，这里是一个时区补偿）。

简单而有效的办法就是按数据源里的时间字符串来定义specifier。

%d和%e区分符在解析时是等同的。

[#](https://github.com/mbostock/d3/wiki/Time-Formatting" \l "wiki-format_multi) d3.time.format.**multi**(*formats*)

返回一个多维的包含预制格式器数组的时间格式器，每个格式器都是一个两元数组组成的format specifier字符串（类似于传递给d3.time.format的构造函数）和一个预制函数。对于传递给返回的时间格式器的任何时间，第一个预制函数返回为true，将决定着指定时间如何被格式化。例如，d3.time.scale使用的是默认时间格式器：

**var** format **=** d3.time.format.multi([ [".%L", **function**(d) { **return** d.getMilliseconds(); }], [":%S", **function**(d) { **return** d.getSeconds(); }], ["%I:%M", **function**(d) { **return** d.getMinutes(); }], ["%I %p", **function**(d) { **return** d.getHours(); }], ["%a %d", **function**(d) { **return** d.getDay() **&&** d.getDate() **!=** 1; }], [

**by JiGuoliang**