### [内部方法（Internals）](https://github.com/mbostock/d3/wiki/Internals)

* [d3.functor](https://github.com/mbostock/d3/wiki/Internals#wiki-functor) - 函数化。将非函数变量转化为只返回该变量值的函数。输入函数，则返回原函数；输入值，则返回一个函数，该函数只返回原值。
* [d3.rebind](https://github.com/mbostock/d3/wiki/Internals#wiki-rebind) - 将一个对象的方法绑定到另一个对象上。
* [d3.dispatch](https://github.com/mbostock/d3/wiki/Internals#wiki-d3_dispatch) - 创建一个定制的事件。
* [dispatch.on](https://github.com/mbostock/d3/wiki/Internals#wiki-dispatch_on) - 添加或移除一个事件监听器。对一个事件可添加多个监听器。
* [dispatch.type](https://github.com/mbostock/d3/wiki/Internals#wiki-_dispatch) - 触发事件。其中‘type’为要触发的事件的名称。

## Functions

[#](https://github.com/mbostock/d3/wiki/Internals" \l "wiki-functor) d3.**functor**(*value*)

如果参数 value 是个函数，返回这个函数。否则，返回一个能够输出这个参数的函数变量。该方法用来将常量参数升级转换成函数，以备需要指定属性为常量或者函数的时候，直接实现。比如：许多D3 layouts 需要指定属性成这种格式，当我们自动转换值到函数的时候，这样可以简化实现。

[#](https://github.com/mbostock/d3/wiki/Internals" \l "wiki-rebind) d3.**rebind**(*target*, *source*, *names…*)

将方法从指定的参数source拷贝到target。当调用target，将相当于调用函数source。注意，传递到target的参数，将传递到source中。target使用source作为this的上下文。如果source返回了source对象，那么相应的target将会返回target对象。否则，target返回source返回的值。方法rebind 允许继承的方法绑定到一个不同对象的子类。

## Events

D3中的行为和高级组件，如 [brush](https://github.com/mbostock/d3/wiki/SVG-Controls#wiki-brush)，使用d3.dispatch 来传递事件消息。

[](http://bl.ocks.org/mbostock/5872848)

对于多关联视图的可视化，d3.dispatch 提供一个方便的轻量级的机制来处理相关联的组件。将代码和d3.dispatch 结合起来，可将涉及的多个事件分离，更好的维护自己的代码。

[#](https://github.com/mbostock/d3/wiki/Internals" \l "wiki-d3_dispatch) d3.**dispatch**(*types…*)

为指定的types创建一个dispatcher对象。每个字符串参数表示一个事件相应，比如："zoom" 和 "change"。返回的对象是一个关联的数组。每个type和一个dispatch 相关联。如果你想为start和end创建一个event dispatcher，可以这样：

**var** dispatch **=** d3.dispatch("start", "end");

然后，你可以访问dispatchers 的属性来获取不同的事件相应属性： dispatch.start and dispatch.end. 如，你可以添加一个事件的监听：

dispatch.on("start", listener);

然后传递事件到所有注册的监听器上：

dispatch.start();

关于如何实现将参数传递到监听器，详见： [dispatch](https://github.com/mbostock/d3/wiki/Internals#wiki-dispatch)。

[#](https://github.com/mbostock/d3/wiki/Internals" \l "wiki-dispatch_on) dispatch.**on**(*type*[, *listener*])

为指定的type添加或删除一个事件监听。其中 type 是一个事件名，如 "start" 或 "end"。函数调用将参数和上下文传递给监听，并触发监听。详见 [dispatch](https://github.com/mbostock/d3/wiki/Internals#wiki-dispatch).

如果出在事件监听注册了某个type，已经存在的监听将被删除，然后注册新的监听。为了注册多个事件监听到同一个type，可以为这个typy提供命名空间，如： "click.foo" 和 "click.bar"。

如果参数中没有监听，则默认为给指定的type设置当前的监听。

[#](https://github.com/mbostock/d3/wiki/Internals" \l "wiki-dispatch) dispatch.**type**(*arguments…*)

 type 方法 (如上文中的 dispatch.start ) 通知并将参数传递给注册的监听。上下文 this 作为注册监听的上下文。例如：通过foo 和 bar参数值触发所有的监听，比如dispatch.call( foo, bar )。因此，你可以传递任何参数到指定的监听器上。通常，我们通过创建一个对象来表示一个事件相应，或者是传递当前的datum ( d ) 和 index ( i )。也可以使用 [call](https://developer.mozilla.org/en/JavaScript/Reference/Global_Objects/Function/Call) 或者 [apply](https://developer.mozilla.org/en/JavaScript/Reference/Global_Objects/Function/Apply)来设置监听器的 "this" 上下文。

举例说明：如果想要为"custom" 事件添加一个"click" 事件，并预置上下文this和参数：

selection.on("click", **function**(d, i) {

dispatch.custom.apply(**this**, arguments);

});