- 如发现翻译不当或有其他问题可以通过以下方式联系译者:
- 邮箱：zhang_tianxu@sina.com
- QQ群：D3数据可视化205076374，大数据可视化436442115

实施可重用组件的各种工具。

函数

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。

d3.**rebind**(*target*, *source*, *names...*)

将名称在*names*中的方法(method)从指定的参数*source*拷贝到*target*，并返回*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。

最新英文内容如下

```
Copies the methods with the specified *names* from *source* to *target*, and returns
*target*. Calling one of the named methods on the target object invokes the same-
named method on the source object, passing any arguments passed to the target
method, and using the source object as the `this` context. If the source method
returns the source object, the target method returns the target object ("setter"
method); otherwise, the target method returns the return value of the source method
("getter" mode). The rebind operator allows inherited methods (mix-ins) to be
rebound to a subclass on a different object.
```

事件

D3中的行为和高级组件，如 brush，使用d3.dispatch 来传递事件消息。

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。

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"事件创建一个event dispatcher，可以这样：

```
vardispatch=d3.dispatch("start","end");
```

然后，你可以访问这些dispatchers来获取不同的事件相应属性，例如 `dispatch.start` 和 `dispatch.end`：

```
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。

dispatch.**on**(*type*[, *listener*])

为指定的*type*添加或删除一个事件监听(*listener*)。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.

如果出在事件监听注册了同一个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"。同样的，可以通过 `dispatch.on(".foo", null)` 来移除某个命名空间内注册的所有监听 (Added by Howard L., 11/Feb/2016)。

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。

dispatch.**type**(*arguments...*)

type 方法 (如上文中的 `dispatch.start`) 通知并将参数*arguments*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 或者 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);
});
```

边城T20140403_guluP20141122