链滴

绘画板 10——多选元素

作者：crick77

原文链接：https://ld246.com/article/1474858633193

来源网站：链滴

github地址： https://github.com/wangyuheng/painter

DEMO地址： http://painter.crick.wang/

多选元素

原理

在选择按钮状态下，可以绘制一个曲线矩形，遍历所有元素判断，如果当前元素在矩形的坐标范围内则元素被选中。

矩形四个点的坐标范围为x,x+width,y,y+height

变更element click事件

选择元素时，需要触发元素选中事件。如果触发click，会增加额外判断，并延时。所以单独抽离pick件，将click中的事件处理指向pick

```
_ele.on("click", function() {
    if (GlobalStatus.isPreFilled()) {
        if ($("#fill_color").hasClass("active")) {
            _ele.fill(GlobalStatus.getFillColor());
            _ele.style("fill-opacity", GlobalStatus.getFillOpacity());
        } else {
            _ele.style("stroke", GlobalStatus.getFontColor());
        }

    } else if (GlobalStatus.isPicked()) {
        _ele.fire("pick");
    } else if (GlobalStatus.isRecycle()) {
        _ele.remove();
    }
});
_ele.on("pick", function() {
    if (_ele.attr("picked")) {
        _ele.attr("picked", null);
        _ele.handleBorder && _ele.handleBorder.hideShade(_ele);
        GlobalStatus.removePicked(_ele);
    } else {
        _ele.attr("picked", true);
        _ele.handleBorder = _ele.handleBorder || new HandleBorder(svgDoc);
        _ele.handleBorder.showShade(_ele);
        GlobalStatus.pushPicked(_ele);

    }
});
```

新增draw.tool.pick.js

新增DrawTool.Pick，操作与Rect类似，唯一的区别在于mouseup时判断元素是否在绘制的虚线矩形

围内，并对应触发pick事件。

```
(function() {

    var parent = null;
    var drawing = false;
    var element = null;
    var startPoint = null;

    function mousedown(event) {
        console.log('pick mousedown');
        drawing = true;
        startPoint = svgDoc.transformPoint(event);
        element = parent.rect(0, 0).fill(GlobalStatus.getFillColor()).style({
            "fill-opacity": GlobalStatus.getFillOpacity(),
            "stroke-dasharray": "13 10"
        }).stroke({
            width: "1",
            color: "grey"
        });
        return false;
    }

    function mousemove(event) {
        console.log('pick mousemove');
        if (drawing) {
            var svgPoint = svgDoc.transformPoint(event);
            var x = svgPoint.x;
            var y = svgPoint.y;

            var newWidth = x - startPoint.x;
            var newHeight = y - startPoint.y;
            var startX = startPoint.x;
            var startY = startPoint.y;
            if (newWidth < 0) {
                startX += newWidth;
            }

            if (newHeight < 0) {
                startY += newHeight;
            }
            newWidth = Math.abs(newWidth);
            newHeight = Math.abs(newHeight);
            element.x(startX).y(startY).width(newWidth).height(newHeight);
        }
        return false;
    };

    function mouseup(event) {
        console.log('pick mouseup ' + element);
        drawing = false;
        if (element.attr("width") > 0) {
            var sx = element.x();
```

```
            var ex = element.x() + element.width();
            var sy = element.y();
            var ey = element.y() + element.height();
            $(GlobalStatus.getAllElements()).each(function() {
                console.log(this.x(), this.y(), sx < this.x() && this.x() < ex && sy < this.y() && this.y()
< ey);
                if (sx < this.x() && this.x() < ex && sy < this.y() && this.y() < ey) {
                    if (!this.attr("picked")) {
                        this.fire("pick");
                    }
                } else if (this.attr("picked")) {
                    this.fire("pick");
                }
            })
        }
        parent.removeElement(element);
        return false;
    }

    var listener = {
        mousedown: mousedown,
        mousemove: mousemove,
        mouseup: mouseup,
    };


    var Pick = function(parentEle) {
        parent = parentEle;
        console.log(parent);
        svgDoc = parent.doc();
        DrawTool.init(svgDoc, listener);
        this.stop = function() {
            DrawTool.stop(svgDoc, listener);
        };
    };

    this.DrawTool.Pick = Pick;

})();
```

首页监听选择按钮

在首页监听选择按钮，被选中时，创建DrawTool.Pick对象

```
$("#tool_pick").on("click", function() {
    resetCurrentDrawTool();
    currentDrawTool = new DrawTool.Pick(svgDoc);
});
```

弊端

有一个不足的地方，元素必须全在范围内，才能被选中。如果被选中部分，则无法选中。没想到好的

决方案。

bug修复

picked状态混乱

click和mouseup互相冲突，导致选中状态丢失。

将pick拆分为pick和unPick两个事件，分别处理选中状态，而不进行判断。将判断交给上层调用方法

```
_ele.on("click", function() {
    console.log("click");
    if (GlobalStatus.isPreFilled()) {
        if ($("#fill_color").hasClass("active")) {
            _ele.fill(GlobalStatus.getFillColor());
            _ele.style("fill-opacity", GlobalStatus.getFillOpacity());
        } else {
            _ele.style("stroke", GlobalStatus.getFontColor());
        }

    } else if (GlobalStatus.isPicked()) {
        if (_ele.attr("picked")) {
            _ele.fire("unPick");
        } else {
            _ele.fire("pick");
        }

    } else if (GlobalStatus.isRecycle()) {
        _ele.remove();
    }
});
_ele.on("pick", function() {
    console.log("pick");
    _ele.attr("picked", true);
    _ele.handleBorder = _ele.handleBorder || new HandleBorder(svgDoc);
    _ele.handleBorder.showShade(_ele);
    GlobalStatus.pushPicked(_ele);
});
_ele.on("unPick", function() {
    console.log("unPick");
    _ele.attr("picked", null);
    _ele.handleBorder && _ele.handleBorder.hideShade(_ele);
    GlobalStatus.removePicked(_ele);

});
```

Pick中

```
function mouseup(event) {
    console.log('pick mouseup ' + element);
    if (drawing) {
```

```
        drawing = false;
        if (element && element.attr("width") > 20) {
            var sx = element.x();
            var ex = element.x() + element.width();
            var sy = element.y();
            var ey = element.y() + element.height();
            $(GlobalStatus.getAllElements()).each(function() {
                console.log(this.x(), this.y(), sx < this.x() && this.x() < ex && sy < this.y() && this.y()
< ey);
                if (sx < this.x() && this.x() < ex && sy < this.y() && this.y() < ey) {
                    if (!this.attr("picked")) {
                        this.fire("pick");
                    }
                } else if (this.attr("picked")) {
                    this.fire("unPick");
                }
            })
        }
        element && element.remove();
    }
    return false;
}
```

Pick拥有背景色

修改Pick的mousedown方法，独立设置style，不关联颜色选择器

```
function mousedown(event) {
    console.log('pick mousedown');
    if (!drawing) {
        drawing = true;
        startPoint = svgDoc.transformPoint(event);
        element = parent.rect(0, 0).style({
            "fill-opacity": "0.0",
            "stroke-dasharray": "10"
        }).stroke({
            width: "1",
            color: "grey"
        });
    }
    return false;
}
```

mousedown状态鼠标移出画板范围，松开鼠标，再次回到画板范围内，导致态丢失

再回到画板时，鼠标已经移开，但是并未执行mouseup方法，因为时间的监听范围为画板内。所以mouseover事件继续执行，导致多生成了一个element。为了避免此问题，在mouse事件中增加drawin状态判断，以Rect为例

```
(function() {
```

```
    var parent = null;
    var drawing = false;
    var element = null;
    var startPoint = null;

    function mousedown(event) {
        console.log('rect mousedown');
        if (!drawing) {
            drawing = true;
            startPoint = svgDoc.transformPoint(event);
            element = parent.rect(0, 0).fill(GlobalStatus.getFillColor()).style("fill-opacity", GlobalSta
us.getFillOpacity()).stroke({
                width: GlobalStatus.getLineSize(),
                color: GlobalStatus.getFontColor()
            });
        }
        return false;
    }

    function mousemove(event) {
        console.log('rect mousemove');
        if (drawing) {
            var svgPoint = svgDoc.transformPoint(event);
            var x = svgPoint.x;
            var y = svgPoint.y;

            var newWidth = x - startPoint.x;
            var newHeight = y - startPoint.y;
            var startX = startPoint.x;
            var startY = startPoint.y;
            if (newWidth < 0) {
                startX += newWidth;
            }

            if (newHeight < 0) {
                startY += newHeight;
            }
            newWidth = Math.abs(newWidth);
            newHeight = Math.abs(newHeight);
            element.x(startX).y(startY).width(newWidth).height(newHeight);
        }
        return false;
    };

    function mouseup(event) {
        console.log('rect mouseup ' + element);
        if (drawing) {
            drawing = false;
            if (element.attr("width") > 0) {
                element.pickable();
            } else {
                parent.removeElement(element);
            }
```

```
        }
        return false;
    }

    var listener = {
        mousedown: mousedown,
        mousemove: mousemove,
        mouseup: mouseup,
    };


    var Rect = function(parentEle) {
        parent = parentEle;
        console.log(parent);
        svgDoc = parent.doc();
        DrawTool.init(svgDoc, listener);
        this.stop = function() {
            DrawTool.stop(svgDoc, listener);
        };
    };

    this.DrawTool.Rect = Rect;

})();
```

所有DrawTool类方法都要增加此

bug修复

选择其他DrawTool后，选中状态不丢失。

在GlobalStatus增加清除所有选中状态的方法

```
unPickAll() {
    $(GlobalStatus.getPickeds()).each(function() {
        this.fire("unPick");
    });
    return this;
}
```

在index的resetCurrentDrawTool方法中调用执行

```
function resetCurrentDrawTool() {
    currentDrawTool && currentDrawTool.stop();
    GlobalStatus.unPickAll();
    $("#svgPanel").css("cursor", "default");
}
```

同时给右键的取消按钮增加此方法调用

```
label: ' 取消',
action: function () {
    GlobalStatus.unPickAll();
    return false;
}
```