

# Getting Start

## 디펜던시

오픈그래프의 디펜던시는 Jquery, Jquery UI, Jquery context menu 입니다.

- [Jquery \(https://jquery.com/\)](https://jquery.com/)
- [Jquery UI \(https://jqueryui.com/\)](https://jqueryui.com/)
- [Jquery context menu \(http://swisnl.github.io/jQuery-contextMenu/index.html\)](http://swisnl.github.io/jQuery-contextMenu/index.html)

## 인스톨과 시작 예제

### html 준비

먼저 빈 html 페이지 하나를 준비합니다.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ko" lang="ko">
<head>
    <title>BPMN Modeler Example</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
    <meta http-equiv="X-UA-Compatible" content="IE=EmulateIE8">

</head>
<body>

</body>
</html>
```

### 라이브러리 인스톨

다음은 오픈그래프에서 기본적으로 제공하는 Symbol 이미지들과 디펜던시들을 예제 html 이 불러올 수 있도록 해야 합니다.

본 프로젝트의 webapp/examples 하위에 위치한 resources/images/symbol 와 webapp/examples/lib 폴더 하위에 위치한 opengraph,jquery,jquery-ui,contextmenu 폴더들을 찾도록 합니다.

이 폴더들을 예제 html 과 동일한 디렉토리 구조(또는 url 패스)로 복사하도록 합니다.

```
--- /a directory/resources/images/symbol
    /lib/opengraph
    /lib/jquery
    /lib/jquery-ui
    /lib/contextmenu
    /your-html-file
```

위에서 언급한 디펜던시 파일들과 오픈그래프 라이브러리 js 를 예제 html 로 임포트합니다.

임포트를 완료한 페이지의 샘플 코드는 다음과 같습니다.

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ko" lang="ko">
<head>
    <title>BPMN Modeler Example</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
    <meta http-equiv="X-UA-Compatible" content="IE=EmulateIE8">

    <!-- jquery -->
    <script type="text/javascript" src=".lib/jquery-1.11.1/jquery-
1.11.1.min.js"></script>

    <!-- jquery ui -->
    <script type="text/javascript" src=".lib/jquery-ui-1.11.0.custom/jquery-
ui.min.js"></script>
    <link rel="stylesheet" type="text/css" href=".lib/jquery-ui-
1.11.0.custom/jquery-ui.css"/>

    <!-- jquery Context Menu -->
    <link rel="stylesheet" type="text/css"
href=".lib/contextmenu/jquery.contextMenu.css"/>
    <script type="text/javascript" src=".lib/contextmenu/jquery.contextMenu-
min.js"></script>

    <!-- Opengraph -->
    <script type="text/javascript" src=".lib/opengraph/OpenGraph-0.1.1-
SNAPSHOT.js"></script>

</head>
<body>

</body>
</html>

```

## 캔버스 렌더링

계속해서, 시작 예제를 그릴 캔버스 Div 하나를 화면에 선언합니다.

```

<body>

<div id="canvas" style="cursor: default;"></div>

</body>

```

선언한 div 를 오픈그래프 캔버스로 렌더링합니다.

`new OG.Canvas(div 아이디, 캔버스 사이즈, 백그라운드 컬러, 백그라운드 이미지)` 메소드를 호출함으로써 캔버스를 렌더링 할 수 있습니다.

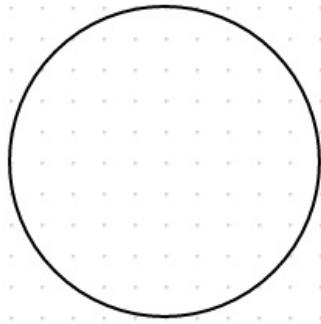
```
<script type="text/javascript">
    $(document).ready(function () {
        var canvas = new OG.Canvas('canvas', [1000, 800], 'white',
'url(resources/images/symbol/grid.gif)');
    });
</script>
```

## 도형 그리기

원형의 도형을 캔버스에 하나 그려보도록 합니다.

canvas.drawShape(좌표[x,y], 도형 , 도형 사이즈[가로,세로]);

```
<script type="text/javascript">
    $(document).ready(function () {
        var canvas = new OG.Canvas('canvas', [1000, 800], 'white',
'url(resources/images/symbol/grid.gif)');
        var circleShape = canvas.drawShape([100, 100], new
OG.CircleShape(), [100, 100]);
    });
</script>
```



여기서 캔버스의 `drawShape` 메소드를 통해서 도형을 그릴때, 그려야 할 도형으로 `new OG.CircleShape()` 를 선언하였습니다.

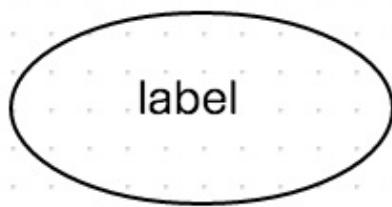
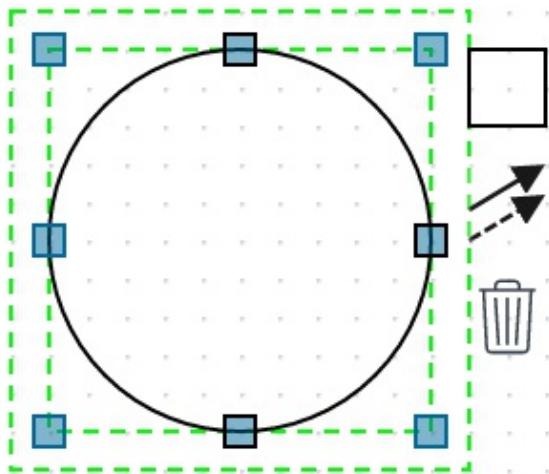
Shape 은 오픈그래프에서 기본 제공하는 Shape 들 이외에도 사용자가 직접 커스터마이징 하여 정의할 수 있습니다.

Shape 에 대한 종류와 커스터마이징에 대해서는 [Shape \(shape.md\)](#) 페이지를 참조하십시오.

## 선 연결

오픈그래프의 도형들은 기본적으로 UI 상에서 Copy, 선 연결, 삭제 할 수 있는 아이콘을 제공하고, 사이즈 재조정 및 이동이 가능합니다.

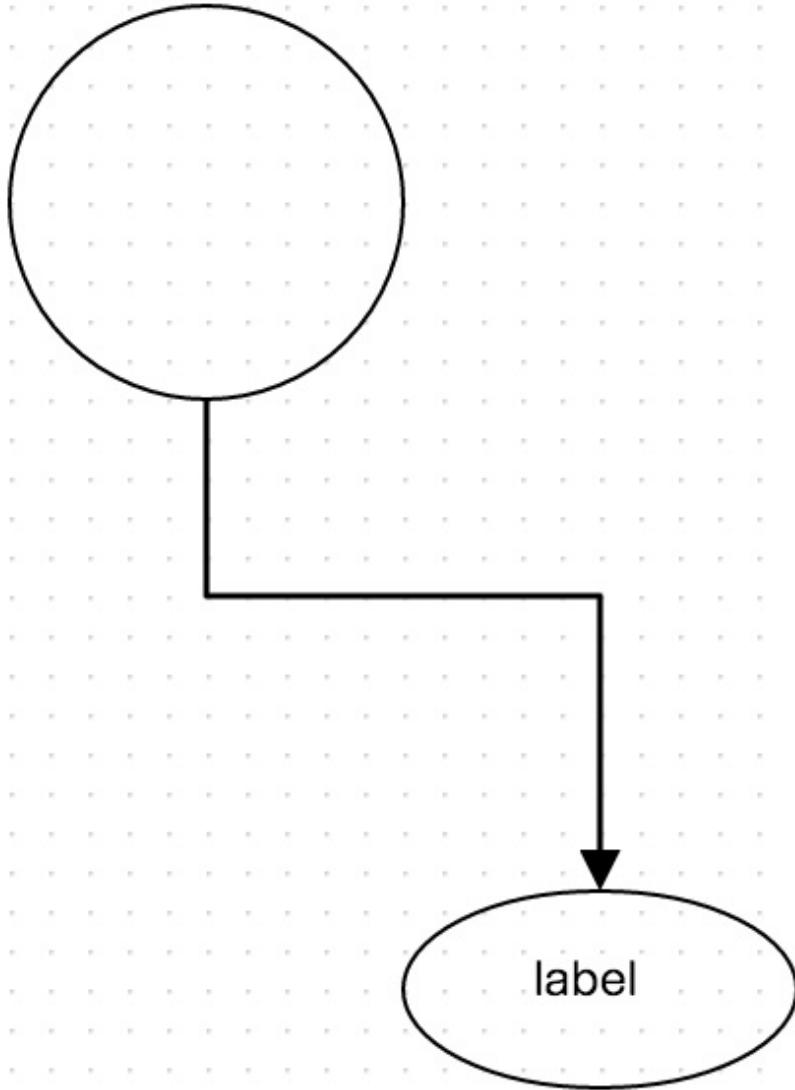
처음 `circleShape` 를 클릭하여 보면 다음의 아이콘들이 나타납니다.



두 개의 도형을 연결하기 위해 하나의 도형을 캔버스에 더 그려보도록 합니다.

```
<script type="text/javascript">
    $(document).ready(function () {
        var canvas = new OG.Canvas('canvas', [1000, 800], 'white',
'url(resources/images/symbol/grid.gif)');
        var circleShape = canvas.drawShape([100, 100], new
OG.CircleShape(), [100, 100]);
        var ellipseShape = canvas.drawShape([200, 300], new
OG.EllipseShape('label'), [100, 50]);
    });
</script>
```

처음 circleShape 의 클릭하여 선연결 모양의 아이콘을 클릭 또는 드래그 하여, ellipseShape 를 클릭하거나 마우스 업을 실행하여 보십시오.



위와 같은 이벤트를 프로그램적으로 처리할 수 있습니다.

canvas.connect(선행 도형, 후행 도형);

```
<script type="text/javascript">
    $(document).ready(function () {
        var canvas = new OG.Canvas('canvas', [1000, 800], 'white',
'url(resources/images/symbol/grid.gif)');
        var circleShape = canvas.drawShape([100, 100], new
OG.CircleShape(), [100, 100]);
        var ellipseShape = canvas.drawShape([200, 300], new
OG.EllipseShape('label'), [100, 50]);
        var edge = canvas.connect(circleShape, ellipseShape);
    });
</script>
```

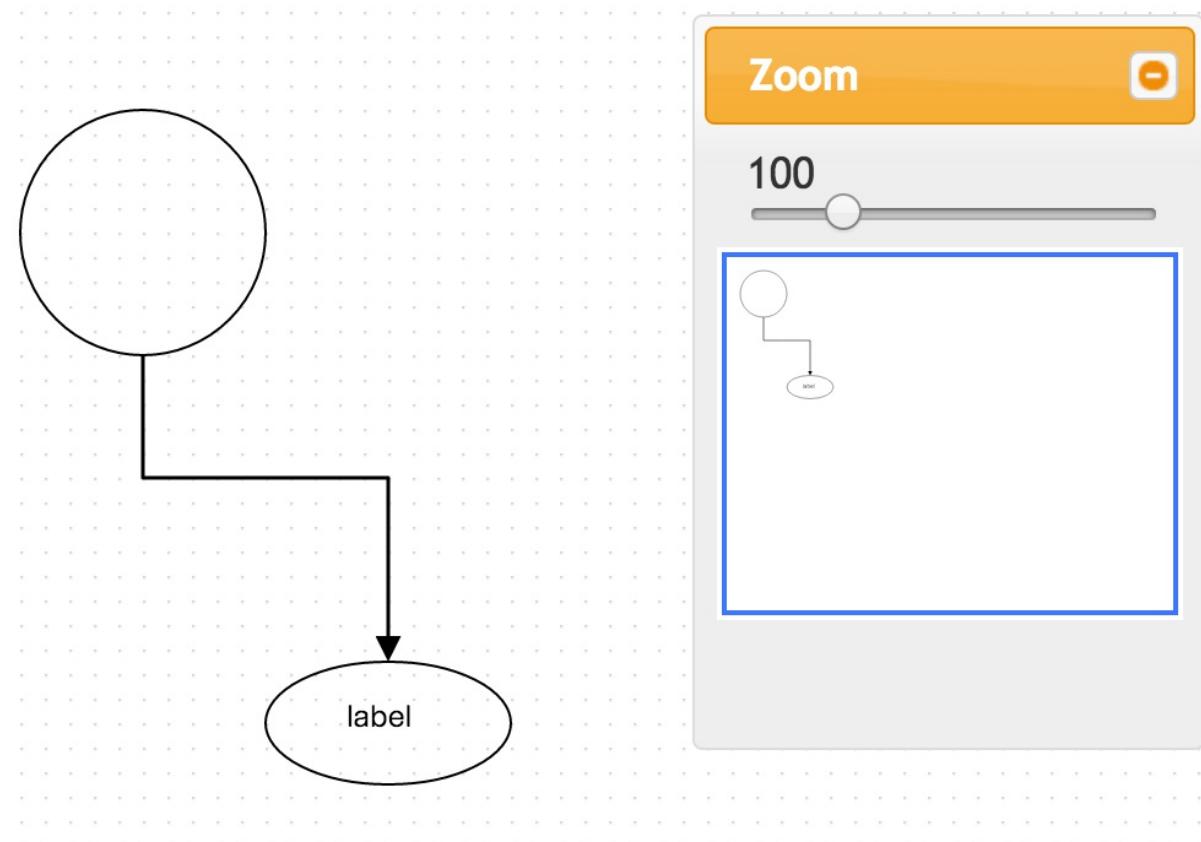
## 줌 패널

줌 패널을 추가하기 위해서는 줌 패널로 사용할 div 를 하나 선언합니다.

```
<body>  
  
<div id="canvas" style="cursor: default;"></div>  
<div id="canvas_slider"></div>  
</body>
```

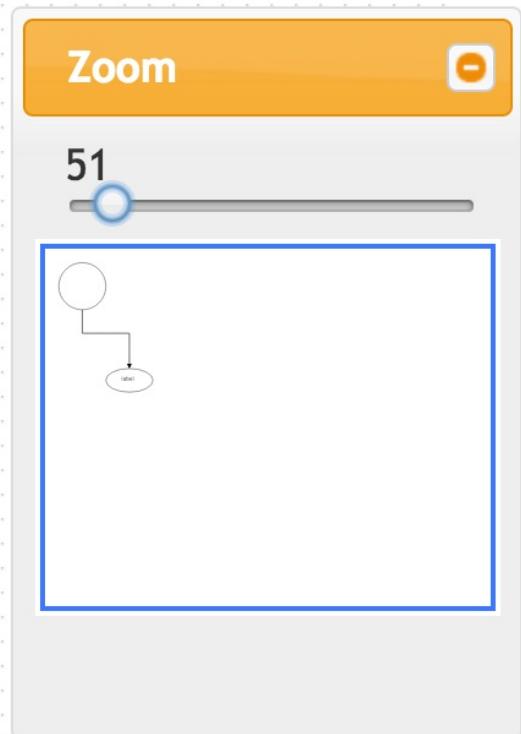
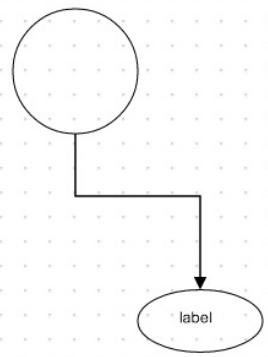
이후에 스크립트에서 슬라이더를 추가하는 명령어를 수행해줍니다.

```
<script type="text/javascript">  
    $(document).ready(function () {  
        var canvas = new OG.Canvas('canvas', [1000, 800], 'white',  
            'url(resources/images/symbol/grid.gif)');  
        canvas.addSlider({  
            slider: $("#canvas_slider"),  
            width: 200,  
            height: 300,  
            appendTo: "body"  
        });  
  
        var circleShape = canvas.drawShape([100, 100], new  
            OG.CircleShape(), [100, 100]);  
        var ellipseShape = canvas.drawShape([200, 300], new  
            OG.EllipseShape('label'), [100, 50]);  
        var edge = canvas.connect(circleShape, ellipseShape);  
    });  
</script>
```



줌 패널의 기능을 `canvas.setScale(0 ~ 1)` 을 통해 프로그램적으로 수행할 수 있습니다.

```
<script type="text/javascript">
    $(document).ready(function () {
        var canvas = new OG.Canvas('canvas', [1000, 800], 'white',
        'url(resources/images/symbol/grid.gif)');
        canvas.addSlider({
            slider: $("#canvas_slider"),
            width: 200,
            height: 300,
            appendTo: "body"
        });
        canvas.setScale(0.51);
    });
</script>
```



# Canvas Drawing

- [Configuration](#)
- [More Detail Configuration](#)
- [Zoom Panel](#)
- [drawShape](#)
- [style](#)
- [label style](#)
- [style](#)
- [set Id](#)
- [set Parent](#)
- [setShapeStyle](#)
- [setTextListInController](#)
- [drawLabel](#)
- [connect](#)
- [disconnect](#)
- [group](#)
- [clear](#)
- [removeShape](#)
- [removeChild](#)
- [getRootElement](#)
- [getRootGroup](#)
- [attribute](#)
- [front and back](#)
- [canvas size](#)
- [scale](#)
- [show and hide](#)
- [append and insert](#)
- [move](#)
- [rotate](#)
- [resize](#)
- [align](#)
- [undo && redo](#)
- [getBoundary](#)

오픈그래프의 캔버스 객체는 다음을 선언함으로써 얻을 수 있습니다.

```
var canvas = new OG.Canvas('canvas', [1000, 800], 'white',
    'url(resources/images/symbol/grid.gif)');
```

## Configuration

오픈그래프 캔버스의 Configuration 을 알아보겠습니다.

* - selectable	: 클릭선택 가능여부(디폴트 true)
* - dragSelectable	: 마우스드래그선택 가능여부(디폴트 true)
* - movable	: 이동 가능여부(디폴트 true)
* - resizable	: 리사이즈 가능여부(디폴트 true)
* - connectable	: 연결 가능여부(디폴트 true)
* - selfConnectable	: Self 연결 가능여부(디폴트 true)
* - connectCloneable	: 드래그하여 연결시 대상 없을 경우 자동으로 Shape 복사하여 연결 처리 여부(디폴트 true)
* - connectRequired	: 드래그하여 연결시 연결대상 있는 경우에만 Edge 드로잉 처리 여부(디폴트 true)
* - labelEditable	: 라벨 수정여부(디폴트 true)
* - groupDropable	: 그룹핑 가능여부(디폴트 true)
* - enableHotKey	: 핫키 가능여부(디폴트 true)
* - enableContextMenu	: 마우스 우클릭 메뉴 가능여부(디폴트 true)
* - autoExtensional	: 캔버스 자동 확장 기능(디폴트 true)
* - useSlider	: 확대축소 슬라이더 사용 여부
* - stickGuide	: 스틱 가이드 표시 여부
* - checkBridgeEdge	: 연결된 두 오브젝트의 소속에 따른 연결선 스타일 변화 여부

Configuration 설정은 canvas 를 선언한 후 initConfig 메소드로 조정할 수 있습니다.

```
canvas.initConfig({
    selectable: true,
    dragSelectable: true,
    movable: true,
    resizable: true,
    connectable: true,
    selfConnectable: true,
    connectCloneable: true,
    connectRequired: true,
    labelEditable: true,
    groupDropable: true,
    enableHotKey: true,
    enableContextMenu: true,
    useSlider: true,
    stickGuide: true,
    checkBridgeEdge: true
});
```

예를 들어, 도형의 이동 및 사이즈 조절이 불가능한 캔버스를 만들고싶다면, 다음과 같은 코드로 생성할 수 있습니다.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ko" lang="ko">
<head>
    <title>BPMN Modeler Example</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
    <meta http-equiv="X-UA-Compatible" content="IE=EmulateIE8">

    <!-- jquery -->
    <script type="text/javascript" src=".//lib/jquery-1.11.1/jquery-
```

```

1.11.1.min.js"></script>

    <!-- jquery ui -->
    <script type="text/javascript" src=".lib/jquery-ui-1.11.0.custom/jquery-
ui.min.js"></script>
    <link rel="stylesheet" type="text/css" href=".lib/jquery-ui-
1.11.0.custom/jquery-ui.css"/>

    <!-- jquery Context Menu -->
    <link rel="stylesheet" type="text/css"
href=".lib/contextmenu/jquery.contextMenu.css"/>
    <script type="text/javascript" src=".lib/contextmenu/jquery.contextMenu-
min.js"></script>

    <!-- Opengraph -->
    <script type="text/javascript" src=".lib/opengraph/OpenGraph-0.1.1-
SNAPSHOT.js"></script>

<script type="text/javascript">
    $(document).ready(function () {
        var canvas = new OG.Canvas('canvas', [1000, 800], 'white',
'url(resources/images/symbol/grid.gif)');
        canvas.initConfig({
            selectable: true,
            dragSelectable: true,
            movable: true,
            resizable: true,
            connectable: true,
            selfConnectable: true,
            connectCloneable: true,
            connectRequired: true,
            labelEditable: true,
            groupDropable: true,
            collapsible: true,
            enableHotKey: true,
            enableContextMenu: true,
            useSlider: true,
            stickGuide: true,
            checkBridgeEdge: true
        });

        //Simple example
        var circleShape = canvas.drawShape([100, 100], new
OG.CircleShape(), [100, 100]);
        var ellipseShape = canvas.drawShape([200, 300], new
OG.EllipseShape('label'), [100, 50]);
        var edge = canvas.connect(circleShape, ellipseShape);
    });

</script>
</head>
<body>

```

```
<div id="canvas" style="cursor: default;"></div>
<div id="canvas_slider"></div>

</body>
</html>
```

## More Detail Configuration

더 자세한 Configuration 설정값에 대해 알아보겠습니다.

캔버스 객체의 \_CONFIG 오브젝트 값을 직접 설정함으로써 Configuration 을 바꿀 수 있습니다.

예를 들면, 선 연결이 불가능한 캔버스를 \_CONFIG 오브젝트 설정으로 만들고싶을 때, CONNECTABLE 값을 false 로 지정함으로써 가능합니다.

```
var canvas = new OG.Canvas('canvas', [1000, 800], 'white',
  'url(resources/images/symbol/grid.gif)');
canvas._CONFIG.CONNECTABLE = false;
```

아래의 목록은 \_CONFIG 를 통해 조절할 수 있는 Configuration 의 목록입니다.

```
{
  /**
   * 도형 선택시 캔버스 포거싱 여부
   */
  FOCUS_CANVAS_ONSELECT: true,

  /**
   * 연결된 두 오브젝트의 소속에 따른 연결선 스타일 변화 여부
   */
  CHECK_BRIDGE_EDGE: true,

  /**
   * 스틱 가이드 생성 여부
   */
  STICK_GUIDE: true,

  /**
   * 슬라이더
   */
  SLIDER: null,

  /**
   * 서버 수신 데이터 처리중
   */
  REMOTE_PERFORMED_DURING: false,

  /**
   * 리모트 데피니션
   */
}
```

```
REMOTE_IDENTIFIER: null,  
  
/**  
 * 리모트 모드  
 */  
REMOTEABLE: false,  
  
/**  
 * 리모트 모드 수정권한  
 */  
REMOTE_EDITABLE: false,  
  
/**  
 * 리모트 모드 마스터모드  
 */  
REMOTE_ISMASTER: false,  
  
/**  
 * 히스토리 인덱스  
 */  
HISTORY_INDEX: 0,  
  
/**  
 * 히스토리 저장소  
 */  
HISTORY: [],  
  
/**  
 * 히스토리 저장 횟수  
 */  
HISTORY_SIZE: 100,  
  
/**  
 * 확대/축소 슬라이더  
 */  
USE_SLIDER: true,  
  
/**  
 * 클릭선택 가능여부  
 */  
SELECTABLE: true,  
  
/**  
 * 마우스드래그선택 가능여부  
 */  
DRAG_SELECTABLE: true,  
  
/**  
 * 이동 가능여부  
 */  
MOVABLE: true,  
MOVABLE_: {
```

```
    GEOM: true,
    TEXT: true,
    HTML: true,
    IMAGE: true,
    EDGE: true,
    GROUP: true
  },
  /**
   * 리사이즈 가능여부
   */
  RESIZABLE: true,
  RESIZABLE_: {
    GEOM: true,
    TEXT: true,
    HTML: true,
    IMAGE: true,
    EDGE: true,
    GROUP: true
  },
  /**
   * 연결 가능여부
   */
  CONNECTABLE: true,
  /**
   * Self 연결 가능여부
   */
  SELF_CONNECTABLE: true,
  /**
   * 가이드에 자기자신을 복사하는 컨트롤러 여부.
   */
  CONNECT_CLONEABLE: true,
  /**
   * 드래그하여 연결시 연결대상 있는 경우에만 Edge 드로잉 처리 여부
   */
  CONNECT_REQUIRED: true,
  /**
   * 드래그하여 연결시 그룹을 건너뛸때 스타일 변경 여부
   */
  CONNECT_STYLE_CHANGE: true,
  CONNECT_STYLE_CHANGE_: {
    GEOM: true,
    TEXT: true,
    HTML: true,
    IMAGE: true,
    EDGE: true,
```

```
        GROUP: true
    },

    /**
     * 가이드에 삭제 컨트롤러 여부
     */
    DELETABLE: true,
    DELETABLE_: {
        GEOM: true,
        TEXT: true,
        HTML: true,
        IMAGE: true,
        EDGE: true,
        GROUP: true
    },

    /**
     * 라벨 수정여부
     */
    LABEL_EDITABLE: true,
    LABEL_EDITABLE_: {
        GEOM: true,
        TEXT: true,
        HTML: true,
        IMAGE: true,
        EDGE: true,
        GROUP: true
    },

    /**
     * 그룹핑 가능여부
     */
    GROUP_DROPABLE: true,

    /**
     * 이동, 리사이즈 드래그시 MOVE_SNAP_SIZE 적용 여부
     */
    DRAG_GRIDABLE: true,

    /**
     * 핫키 가능여부
     */
    ENABLE_HOTKEY: true,

    /**
     * 핫키 : UNDO REDO 키 가능여부
     */
    ENABLE_HOTKEY_CTRL_Z: true,

    /**
     * 핫키 : DELETE 삭제 키 가능여부
     */
```

```
 */
ENABLE_HOTKEY_DELETE: true,

/**
 * 핫키 : Ctrl+A 전체선택 키 가능여부
 */
ENABLE_HOTKEY_CTRL_A: true,

/**
 * 핫키 : Ctrl+C 복사 키 가능여부
 */
ENABLE_HOTKEY_CTRL_C: true,

/**
 * 핫키 : Ctrl+V 붙여넣기 키 가능여부
 */
ENABLE_HOTKEY_CTRL_V: true,

/**
 * 핫키 : Ctrl+D 복제하기 키 가능여부
 */
ENABLE_HOTKEY_CTRL_D: true,

/**
 * 핫키 : Ctrl+G 그룹 키 가능여부
 */
ENABLE_HOTKEY_CTRL_G: true,

/**
 * 핫키 : Ctrl+U 언그룹 키 가능여부
 */
ENABLE_HOTKEY_CTRL_U: true,

/**
 * 핫키 : 방향키 가능여부
 */
ENABLE_HOTKEY_ARROW: true,

/**
 * 핫키 : Shift + 방향키 가능여부
 */
ENABLE_HOTKEY_SHIFT_ARROW: true,

/**
 * 마우스 우클릭 메뉴 가능여부
 */
ENABLE_CONTEXTMENU: true,

/**
 * 캔버스 스케일(리얼 사이즈 : Scale = 1)
 */

```

```
SCALE: 1,  
  
/**  
 * 캔버스 최소 스케일  
 */  
SCALE_MIN: 0.1,  
  
/**  
 * 캔버스 최대 스케일  
 */  
SCALE_MAX: 10,  
  
/**  
 * Edge 깍은선 패딩 사이즈  
 */  
EDGE_PADDING: 20,  
  
/**  
 * 라벨의 패딩 사이즈  
 */  
LABEL_PADDING: 5,  
  
/**  
 * 라벨 에디터(textarea)의 디폴트 width  
 */  
LABEL_EDITOR_WIDTH: 500,  
  
/**  
 * 라벨 에디터(textarea)의 디폴트 height  
 */  
LABEL_EDITOR_HEIGHT: 16,  
  
/**  
 * 시작, 끝점 라벨의 offsetTop 값  
 */  
FROMTO_LABEL_OFFSET_TOP: 15,  
  
/**  
 * Move & Resize 용 가이드 선 콘트롤 Rect 사이즈  
 */  
GUIDE_LINE_SIZE: 20,  
/**  
 * Move & Resize 용 가이드 선 콘트롤 마진 사이즈  
 */  
GUIDE_LINE_MARGIN: 10,  
  
/**  
 * Move & Resize 용 가이드 콘트롤 Rect 사이즈  
 */  
GUIDE_RECT_SIZE: 8,  
  
/**
```

```
 * Move & Resize 용 가이드 가로, 세로 최소 사이즈
 */
GUIDE_MIN_SIZE: 18,

/**
 * 도형 컨트롤러 아이콘 커넥트 라인 표시 개수
 */
GUIDE_CONTROL_LINE_NUM: 2,

/**
 * Collapse & Expand 용 가이드 Rect 사이즈
 */
COLLAPSE_SIZE: 10,

/**
 * Shape Move & Resize 시 이동 간격
 */
MOVE_SNAP_SIZE: 8,

/**
 * 터미널 cross 사이즈
 */
TERMINAL_SIZE: 5,

/**
 * Shape 복사시 패딩 사이즈
 */
COPY_PASTE_PADDING: 20,

/**
 * Fit Canvas 시 패딩 사이즈
 */
FIT_CANVAS_PADDING: 20,

/**
 * 캔버스 영역 자동 확장 여부
 */
AUTO_EXTENSIONAL: true,

/**
 * 캔버스 영역 자동 확장시 증가 사이즈
 */
AUTO_EXTENSION_SIZE: 100,

/**
 * 캔버스 배경색
 */
CANVAS_BACKGROUND: "#f9f9f9",

/**
 * 이미지 url 정보
 */
```

```
/*
IMAGE_USER: "http://processcodi.com/images/opengraph/User.png",
IMAGE_SEND: "http://processcodi.com/images/opengraph/Send.png",
IMAGE_RECEIVE: "http://processcodi.com/images/opengraph/Receive.png",
IMAGE_MANUAL: "http://processcodi.com/images/opengraph/Manual.png",
IMAGE_SERVICE: "http://processcodi.com/images/opengraph/Service.png",
IMAGE_RULE: "http://processcodi.com/images/opengraph/BusinessRule.png",
IMAGE_SCRIPT: "http://processcodi.com/images/opengraph/Script.png",
IMAGE_MAPPER: "http://processcodi.com/images/opengraph/mapper.png",
IMAGE_WEB: "http://processcodi.com/images/opengraph/w_services.png",

/**
 * Edge 선 자동마춤 각도 최소값
 */
TRIM_EDGE_ANGLE_SIZE: 170,
/**
 * Edge 선 이동딜레이 거리
 */
EDGE_MOVE_DELAY_SIZE: 14,

/**
 * swimLane 리사이즈 최소 폭
 */
LANE_MIN_SIZE: 50,

/**
 * swimLane 확장 기본 폭
 */
LANE_DEFAULT_SIZE: 100,

/**
 * swimLane, pool 생성 기본 가로,세로
 */
POOL_DEFAULT_SIZE: [300, 200],

/**
 * 그룹 하위 shape 와 그룹사이의 여유폭
 */
GROUP_INNER_SAPCE: 10,

/**
 * 라벨 최소 크기(IE)
 */
LABEL_MIN_SIZE: 0,

/**
 * 라벨 최대 크기(IE)
 */
LABEL_MAX_SIZE: 300,
```

```
* 디폴트 스타일 정의
*/
DEFAULT_STYLE: {
    SHAPE: {cursor: "default"},
    GEOM: {
        stroke: "black",
        "fill-r": ".5",
        "fill-cx": ".5",
        "fill-cy": ".5",
        fill: "white",
        "fill-opacity": 0,
        "label-position": "center"
    },
    TEXT: {stroke: "none", "text-anchor": "middle"},
    HTML: {"label-position": "bottom", "text-anchor": "middle", "vertical-align": "top"},  
IMAGE: {"label-position": "bottom", "text-anchor": "middle", "vertical-align": "top"},  
EDGE: {
        stroke: "black",
        fill: "none",
        "fill-opacity": 0,
        "stroke-width": 1.5,
        "stroke-opacity": 1,
        "edge-type": "plain",
        "arrow-start": "none",
        "arrow-end": "block",
        "stroke-dasharray": "",
        "label-position": "center",
        "stroke-linejoin": "round",
        cursor: "pointer"
    },
    EDGE_SHADOW: {
        stroke: "#00FF00",
        fill: "none",
        "fill-opacity": 0,
        "stroke-width": 1,
        "stroke-opacity": 1,
        "arrow-start": "none",
        "arrow-end": "none",
        "stroke-dasharray": "- ",
        "edge-type": "plain",
        cursor: "pointer"
    },
    EDGE_HIDDEN: {
        stroke: "white",
        fill: "none",
        "fill-opacity": 0,
        "stroke-width": 10,
        "stroke-opacity": 0,
        cursor: "pointer"
    }
}
```

```
},
GROUP: {
  stroke: "black",
  fill: "none",
  "fill-opacity": 0,
  "label-position": "bottom",
  "text-anchor": "middle",
  "vertical-align": "top"
},
GROUP_HIDDEN: {stroke: "black", fill: "white", "fill-opacity": 0,
"stroke-opacity": 0, cursor: "move"},
GROUP_SHADOW: {
  stroke: "white",
  fill: "none",
  "fill-opacity": 0,
  "stroke-width": 15,
  "stroke-opacity": 0,
  cursor: "pointer"
},
GROUP_SHADOW_MAPPER: {
  stroke: "white",
  fill: "none",
  "fill-opacity": 0,
  "stroke-width": 1,
  "stroke-opacity": 0,
  cursor: "pointer"
},
GUIDE_BBOX: {
  stroke: "#00FF00",
  fill: "white",
  "fill-opacity": 0,
  "stroke-dasharray": "- ",
  "shape-rendering": "crispEdges",
  cursor: "move"
},
GUIDE_UL: {
  stroke: "#03689a",
  fill: "#03689a",
  "fill-opacity": 0.5,
  cursor: "nwse-resize",
  "shape-rendering": "crispEdges"
},
GUIDE_UR: {
  stroke: "#03689a",
  fill: "#03689a",
  "fill-opacity": 0.5,
  cursor: "nesw-resize",
  "shape-rendering": "crispEdges"
},
GUIDE_LL: {
  stroke: "#03689a",
  fill: "#03689a",
```

```
        "fill-opacity": 0.5,
        cursor: "nesw-resize",
        "shape-rendering": "crispEdges"
    },
    GUIDE_LR: {
        stroke: "#03689a",
        fill: "#03689a",
        "fill-opacity": 0.5,
        cursor: "nwse-resize",
        "shape-rendering": "crispEdges"
    },
    GUIDE_LC: {
        stroke: "#03689a",
        fill: "#03689a",
        "fill-opacity": 0.5,
        cursor: "ew-resize",
        "shape-rendering": "crispEdges"
    },
    GUIDE_UC: {
        stroke: "black",
        fill: "#03689a",
        "fill-opacity": 0.5,
        cursor: "ns-resize",
        "shape-rendering": "crispEdges"
    },
    GUIDE_RC: {
        stroke: "black",
        fill: "#03689a",
        "fill-opacity": 0.5,
        cursor: "ew-resize",
        "shape-rendering": "crispEdges"
    },
    GUIDE_LWC: {
        stroke: "black",
        fill: "#03689a",
        "fill-opacity": 0.5,
        cursor: "ns-resize",
        "shape-rendering": "crispEdges"
    },
    GUIDE_FROM: {stroke: "black", fill: "#00FF00", cursor: "move", "shape-rendering": "crispEdges"},
    GUIDE_TO: {stroke: "black", fill: "#00FF00", cursor: "move", "shape-rendering": "crispEdges"},
    GUIDE_CTL_H: {stroke: "black", fill: "#00FF00", cursor: "ew-resize", "shape-rendering": "crispEdges"},
    GUIDE_CTL_V: {stroke: "black", fill: "#00FF00", cursor: "ns-resize", "shape-rendering": "crispEdges"},
    GUIDE_SHADOW: {stroke: "black", fill: "none", "stroke-dasharray": "- ", "shape-rendering": "crispEdges"},
    GUIDE_LINE: {
        stroke: "black",
```

```
        fill: "none",
        "fill-opacity": 0,
        "stroke-width": 1.2,
        "stroke-opacity": 1,
        "stroke-dasharray": "",
        "arrow-end": "block",
        "stroke-linejoin": "round",
        cursor: "pointer"
    },
    GUIDE_LINE_ESSENSIA: {
        stroke: "black",
        fill: "none",
        "fill-opacity": 0,
        "stroke-width": 1.2,
        "stroke-opacity": 1,
        "stroke-dasharray": "",
        "arrow-start": "diamond",
        "arrow-end": "none",
        "stroke-linejoin": "round",
        cursor: "pointer"
    },
    GUIDE_VIRTUAL_EDGE: {
        stroke: "black",
        fill: "none",
        "fill-opacity": 0,
        "stroke-width": 1,
        "stroke-opacity": 1,
        "stroke-dasharray": "- ",
        "stroke-linejoin": "round",
        "arrow-start": "none",
        "arrow-end": "none"
    },
    GUIDE_LINE_AREA: {
        stroke: "#ffffff",
        fill: "#ffffff",
        "fill-opacity": 0.1,
        "stroke-width": 1,
        "stroke-opacity": 0.2,
        cursor: "pointer"
    },
    GUIDE_RECT_AREA: {
        stroke: "black",
        fill: "#ffffff",
        "fill-opacity": 0,
        "stroke-width": 1,
        "stroke-opacity": 1,
        cursor: "pointer"
    },
    RUBBER_BAND: {stroke: "#0000FF", opacity: 0.2, fill: "#0077FF"},
    DROP_OVER_BBOX: {stroke: "#0077FF", fill: "none", opacity: 0.3, "shape-rendering": "crispEdges"},
}
```

```
LABEL: {"font-size": 12, "font-color": "black", "fill": "none"},  
LABEL_EDITOR: {  
    position: "absolute",  
    overflow: "visible",  
    resize: "none",  
    "text-align": "center",  
    display: "block",  
    padding: 0  
},  
COLLAPSE: {  
    stroke: "black",  
    fill: "none",  
    "fill-opacity": 0,  
    cursor: "pointer",  
    "shape-rendering": "crispEdges"  
},  
COLLAPSE_BBOX: {stroke: "none", fill: "none", "fill-opacity": 0},  
BUTTON: {  
    stroke: "#9FD7FF",  
    fill: "white",  
    "fill-opacity": 0,  
    cursor: "pointer",  
    "shape-rendering": "crispEdges"  
},  
CONNECT_GUIDE_EVENT_AREA: {  
    stroke: "#ffffff",  
    fill: "none",  
    "fill-opacity": 0,  
    "stroke-width": 20,  
    "stroke-opacity": 0  
},  
CONNECT_GUIDE_BBOX: {  
    stroke: "#00FF00",  
    fill: "none",  
    "stroke-dasharray": "- ",  
    "shape-rendering": "crispEdges"  
},  
CONNECT_GUIDE_BBOX_EXPEND: 10,  
CONNECT_GUIDE_SPOT_CIRCLE: {  
    r: 7,  
    stroke: "#A6A6A6",  
    "stroke-width": 1,  
    fill: "#FFE400",  
    "fill-opacity": 0.5,  
    cursor: "pointer"  
},  
CONNECT_GUIDE_SPOT_RECT: {  
    stroke: "#A6A6A6",  
    "stroke-width": 1,  
    fill: "#FFE400",  
    "fill-opacity": 0.2,
```

```

        cursor: "ns-resize",
        w: 20,
        h: 10
    },
    CONNECTABLE_HIGHLIGHT: {
        "stroke-width": 2
    },
    NOT_CONNECTABLE_HIGHLIGHT: {
        fill: "#FAAFBE",
        "fill-opacity": 0.5
    }
}
}
}

```

## Zoom Panel

줌 패널을 생성하고 삭제하는 코드입니다.

```

var canvas = new OG.Canvas('canvas', [1000, 800], 'white',
'url(resources/images/symbol/grid.gif');

//줌 패널 추가
canvas.addSlider({
    slider: $('#canvas_slider'),
    width: 200,
    height: 300,
    appendTo: "body"
});

//줌 패널 삭제
canvas.removeSlider();

```

## drawShape

캔버스에 도형을 위치 및 사이즈 지정하여 드로잉합니다.

메소드의 파라미터는 다음과 같습니다.

- @param {Number[]} position 드로잉할 위치 좌표(중앙 기준)
- @param {OG.shape.IShape} shape Shape
- @param {Number[]} size Shape Width, Height
- @param {OG.geometry.Style|Object} style 스타일 (Optional)
- @param {String} id Element ID 지정 (Optional)
- @param {String} parentId 부모 Element ID 지정 (Optional)
- @param {Boolean} preventDrop Lane, Pool 생성 drop 모드 수행 방지
- @return {Element} Group DOM Element with geometry

## style

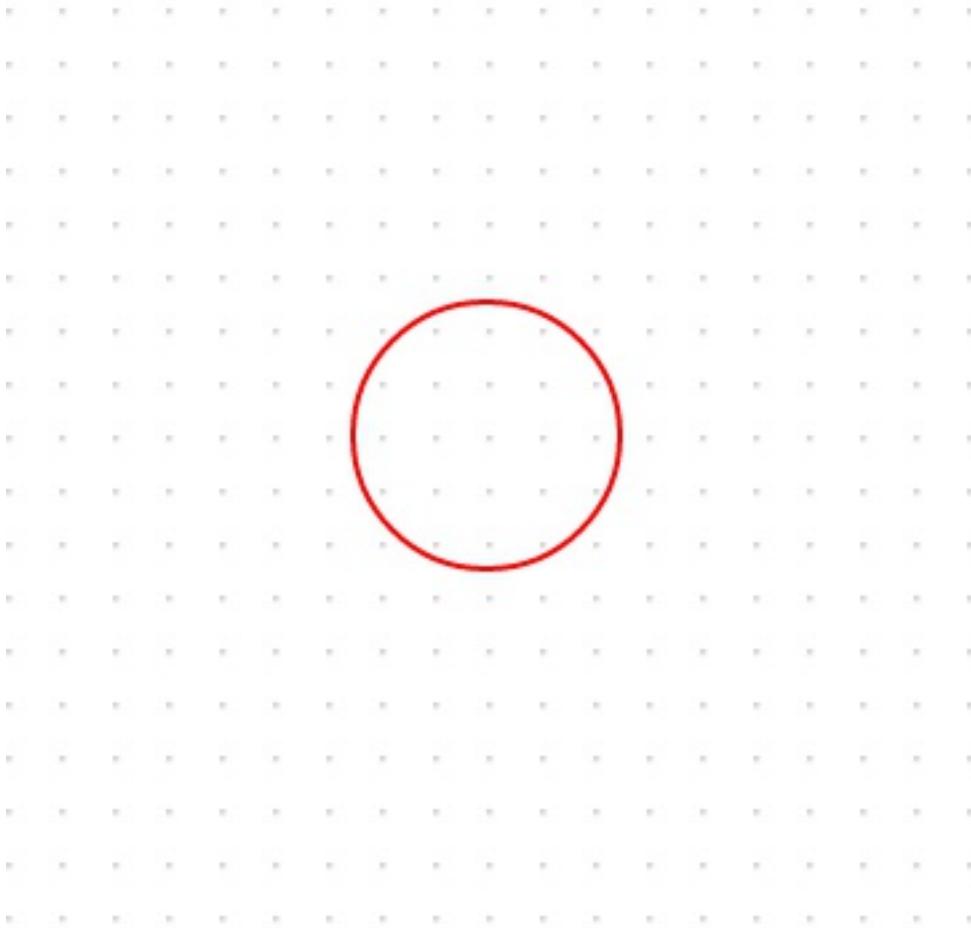
예를 들어 빨간색 테두리의 원 도형을 캔버스에 그릴 때, 다음과 같이 실행합니다.

스타일은 Json 오브젝트 형식으로 지정하며, 일반적인 css 스타일이 아닌 svg 스타일 표준을 따릅니다.

```

var canvas = new OG.Canvas('canvas', [1000, 800], 'white',
'url(resources/images/symbol/grid.gif)');
canvas.drawShape([100, 100], new OG.CircleShape(), [50, 50], {stroke: 'red'});

```



## label style

도형의 라벨 포지션 및 라벨 텍스트 정렬을 위한 오픈그래프 정의 스타일이 있습니다.

스타일 어트리뷰트	값	설명
label-position	top,bottom,left,right,center	도형을 기준으로 라벨의 상,하,좌,우,센터 위치
text-anchor	start,middle,end	라벨 텍스트의 좌,중간,우 정렬
label-direction	vertical,horizontal	라벨 세로 표기,가로 표기
label-width	number	라벨의 길이
vertical-align	top,bottom,middle	라벨의 텍스트의 상단 정렬,하단 정렬
label-fill	string	그룹 도형 한정. 라벨 배경 색상
label-fill-opacity	string	그룹 도형 한정. 라벨 opacity

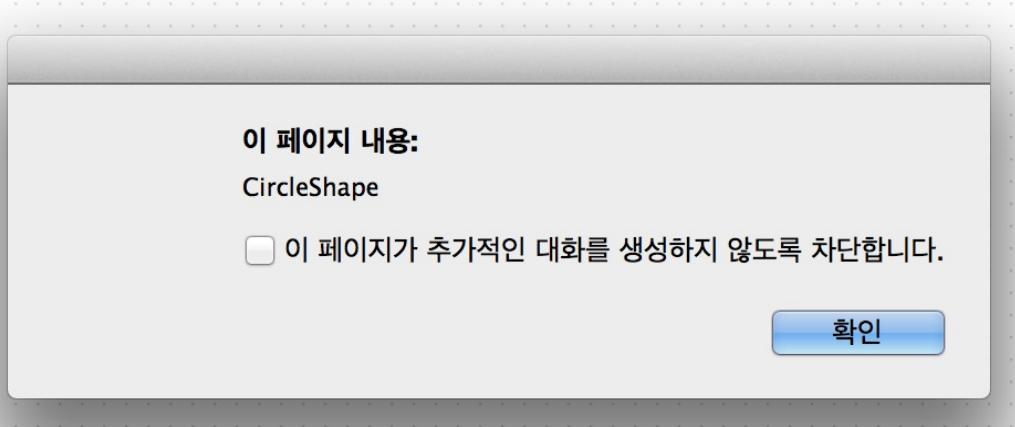
## set Id

도형을 그릴 때 아이디를 부여하지 않으면 랜덤한 아이디가 생성되나, 그릴 때 아이디를 지정하게 되면 해당 도형의 아이디로 지정됩니다.

```

var canvas = new OG.Canvas('canvas', [1000, 800], 'white',
'url(resources/images/symbol/grid.gif)');
var element = canvas.drawShape([100, 100], new OG.CircleShape(), [50, 50],
{stroke: 'red'}, 'CircleShape');
alert(element.id);

```



## set Parent

parentId 를 지정하게 되면 해당 도형은 parentId 가 id 인 도형의 자식으로 그려지게 됩니다.

```

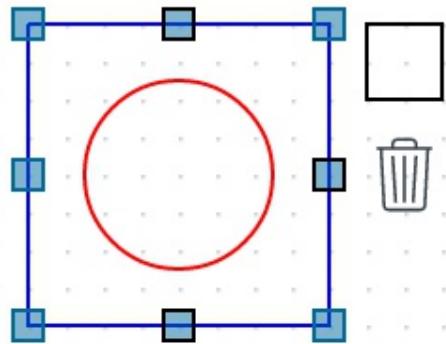
var canvas = new OG.Canvas('canvas', [1000, 800], 'white',
'url(resources/images/symbol/grid.gif)');
var parent = canvas.drawShape([100, 100], new OG.GroupShape(), [80, 80],
{stroke: 'blue'}, 'parentShape');
var element = canvas.drawShape([100, 100], new OG.CircleShape(), [50, 50],
{stroke: 'red'}, 'CircleShape', 'parentShape');

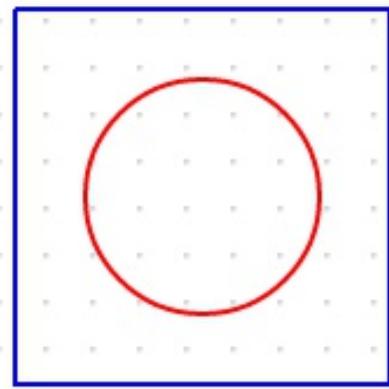
```

이때 parent 가 되는 도형은 GroupShape 이거나 GroupShape 를 상속받는 도형이여야 합니다.

GroupShape 의 자식으로 등록 될 경우 해당 GroupShape 도형을 움직이거나 삭제 또는 복사 할 때, 자식으로 등록된 도형도 함께 적용되게 됩니다.

- 파란색 테두리의 GroupShape 를 이동시 빨간색 CircleShape 가 함께 움직인다.

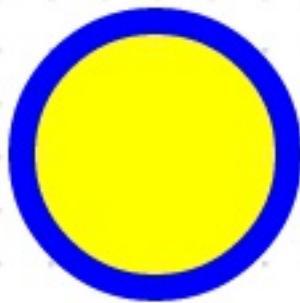




## setShapeStyle

setShapeStyle 메소드를 통해 존재하는 도형의 스타일을 변경 할 수 있습니다.

```
var canvas = new OG.Canvas('canvas', [1000, 800], 'white',
'url(resources/images/symbol/grid.gif)');
var element = canvas.drawShape([100, 100], new OG.CircleShape(), [50, 50],
{stroke: 'red'}, 'CircleShape', 'parentShape');
canvas.setShapeStyle(element, {
  stroke: 'blue',
  'stroke-width': '5',
  fill: 'yellow',
  'fill-opacity': '1'
});
```



## setTextListInController

setTextListInController 메소드를 통해 도형의 연결선에 대한 라벨링을 지정할 수 있습니다.

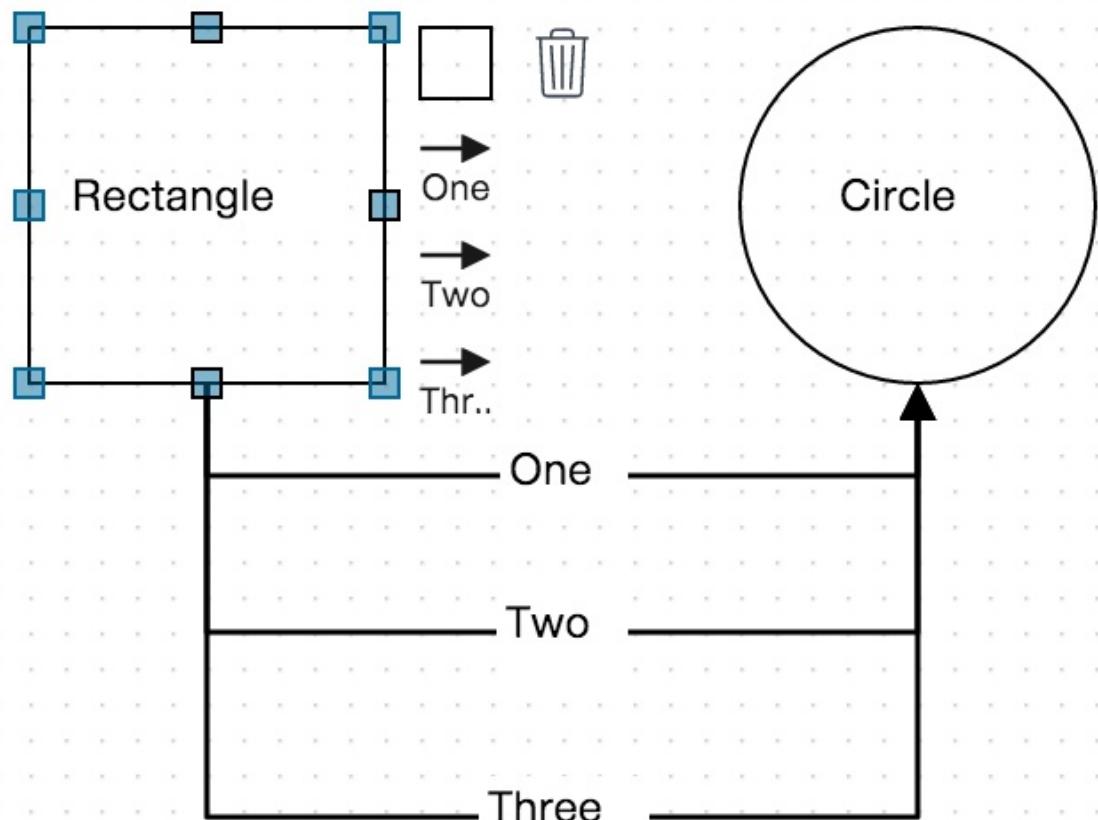
getTextListInController 메소드를 통해 도형의 연결선에 대한 라벨링 리스트를 가져올 수도 있습니다.

```
var canvas = new OG.Canvas('canvas', [1000, 800], 'white',
'url(resources/images/symbol/grid.gif');

var rectangleShape = canvas.drawShape([100, 100], new
OG.RectangleShape('Rectangle'), [100, 100]);
var ellipseShape = canvas.drawShape([300, 100], new OG.CircleShape('Circle'),
[100, 100]);

canvas.setTextListInController(rectangleShape, ['One', 'Two', 'Three']);

console.log(canvas.getTextListInController(rectangleShape));
```



만일, 위의 'One', 'Two', 'Three' 를 선택하여 도형 연결시 연결선을 각기다른 shape 로 표현하기 위해서 다음과 같이 표현하도록 합니다.

## drawLabel

도형에 라벨을 붙일 때는 도형을 정의 할 때 라벨을 정의하는 방법과 canvas 메소드를 통하는 방법이 있습니다.

```
var canvas = new OG.Canvas('canvas', [1000, 800], 'white',
'url(resources/images/symbol/grid.gif');

//case1)
var rectangleShape = canvas.drawShape([100, 100], new
OG.RectangleShape('Rectangle'), [100, 100]);

//case2)
canvas.drawLine(rectangleShape, 'Redraw Label', {'font-size': 30});
```



## connect

두개의 Shape 을 Edge 로 연결합니다.

- @param {Element} fromElement from Shape Element
- @param {Element} toElement to Shape Element
- @param {OG.geometry.Style|Object} style 스타일
- @param {String} label Label
- @param fromP fromElement 와 연결될 터미널 좌표 [x,y \(optional\)](#)
- @param toP toElement 와 연결될 터미널 좌표 [x,y \(optional\)](#)
- @param preventTrigger 참 일 경우 이벤트 발생을 방지
- @param id 연결선의 아이디
- @returns {\*|Element}

연결선과 도형의 접점 지점을 터미널이라 하는데, 이 터미널의 좌표를 따로 설정하지 않으면 터미널은 도형의 센터가 됩니다.

```

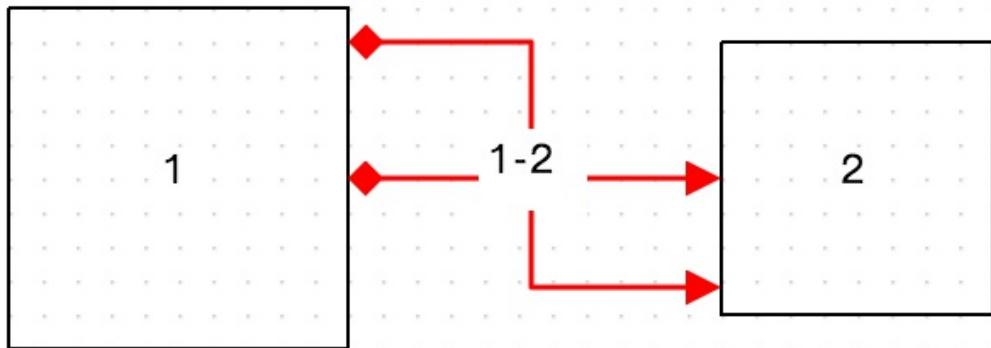
var canvas = new OG.Canvas('canvas', [1000, 800], 'white',
'url(resources/images/symbol/grid.gif');

var rectangleShape = canvas.drawShape([100, 100], new OG.RectangleShape('1'),
[100, 100]);
var circleShape = canvas.drawShape([300, 100], new OG.RectangleShape('2'), [80,
80]);

/**
 * connect at default center position
 */
canvas.connect(rectangleShape, circleShape, {
  stroke: 'red',
  'arrow-start': 'diamond',
  'arrow-end': 'block'
}, '1-2');

/**
 * connect at custom position
 */
canvas.connect(rectangleShape, circleShape, {
  stroke: 'red',
  'arrow-start': 'diamond',
  'arrow-end': 'block'
}, '1-2', [100, 60], [300, 130]);

```



## disconnect

연결속성정보를 삭제합니다. Edge 인 경우는 연결 속성 정보를 삭제하고, 일반 Shape 인 경우는 연결된 모든 Edge 를 삭제합니다.

```

var canvas = new OG.Canvas('canvas', [1000, 800], 'white',
'url(resources/images/symbol/grid.gif');

var rectangleShape = canvas.drawShape([100, 100], new OG.RectangleShape('1'),
[100, 100]);
var circleShape = canvas.drawShape([300, 200], new OG.RectangleShape('2'), [80,
80]);

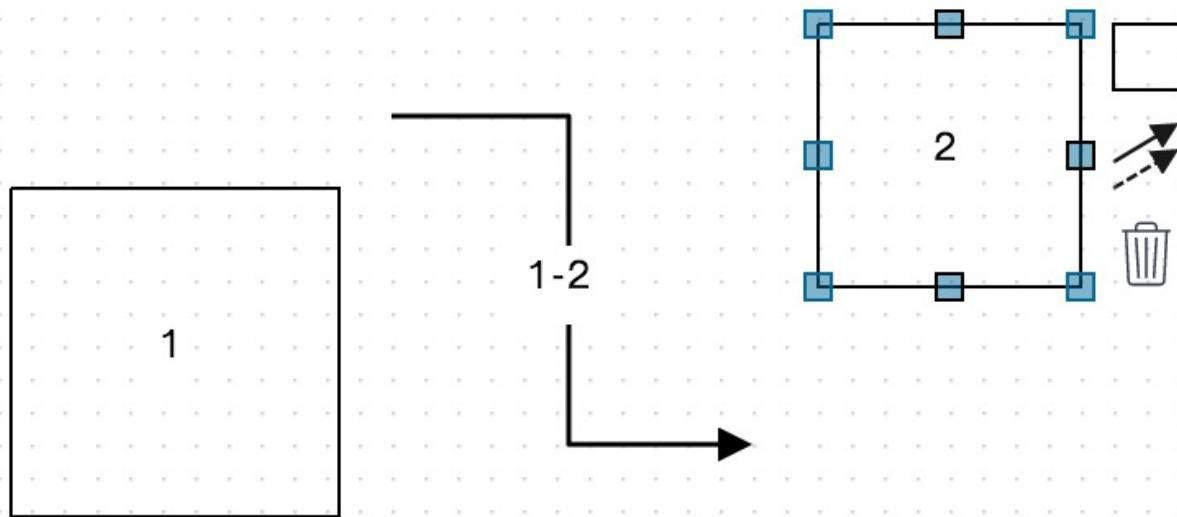
var edge = canvas.connect(rectangleShape, circleShape, null, '1-2');

//case1
canvas.disconnect(edge);

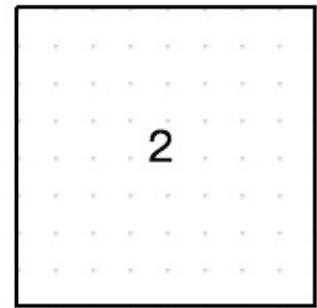
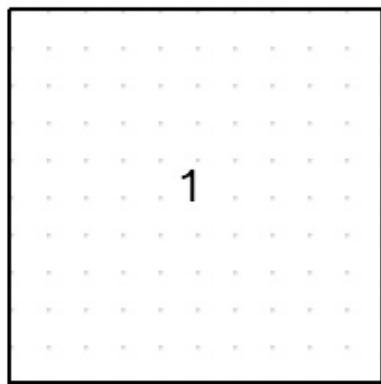
//case2
canvas.disconnect(rectangleShape);

```

- case1) 연결선을 disconnect 한 경우 연결 속성 정보가 사려졌기 때문에 도형을 이동시켜도 재연결 처리가 되지 않는다.



- case2) 도형을 disconnect 한 경우 도형과 연결된 모든 연결선이 삭제된다.



## group

주어진 도형들을 그룹핑합니다. 이때, 연결선은 그룹핑 하는 대상에 넣지 않도록 합니다.

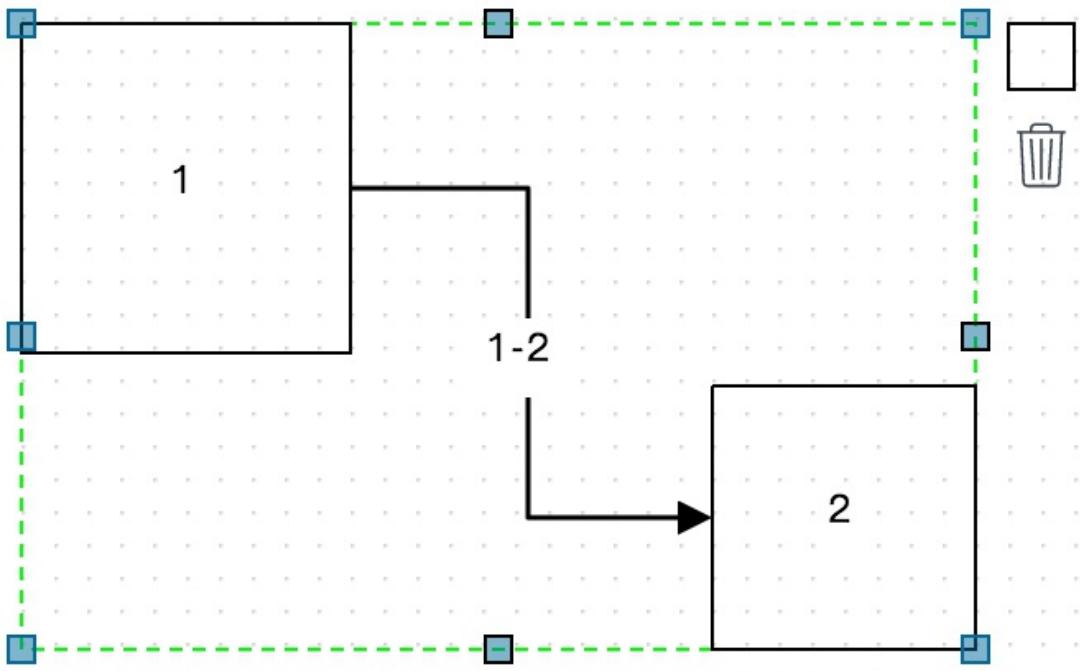
그룹핑의 대상은 연결선이 아닌 도형들입니다.

```
var canvas = new OG.Canvas('canvas', [1000, 800], 'white',
'url(resources/images/symbol/grid.gif');

var rectangleShape = canvas.drawShape([100, 100], new OG.RectangleShape('1'),
[100, 100]);
var circleShape = canvas.drawShape([300, 200], new OG.RectangleShape('2'), [80,
80]);

var edge = canvas.connect(rectangleShape, circleShape, null, '1-2');

//Do not insert edge in array
canvas.group([rectangleShape,circleShape]);
```



그룹핑을 해제하는 예제입니다.

```
var group = canvas.group([rectangleShape,circleShape]);
canvas.ungroup([group]);
```

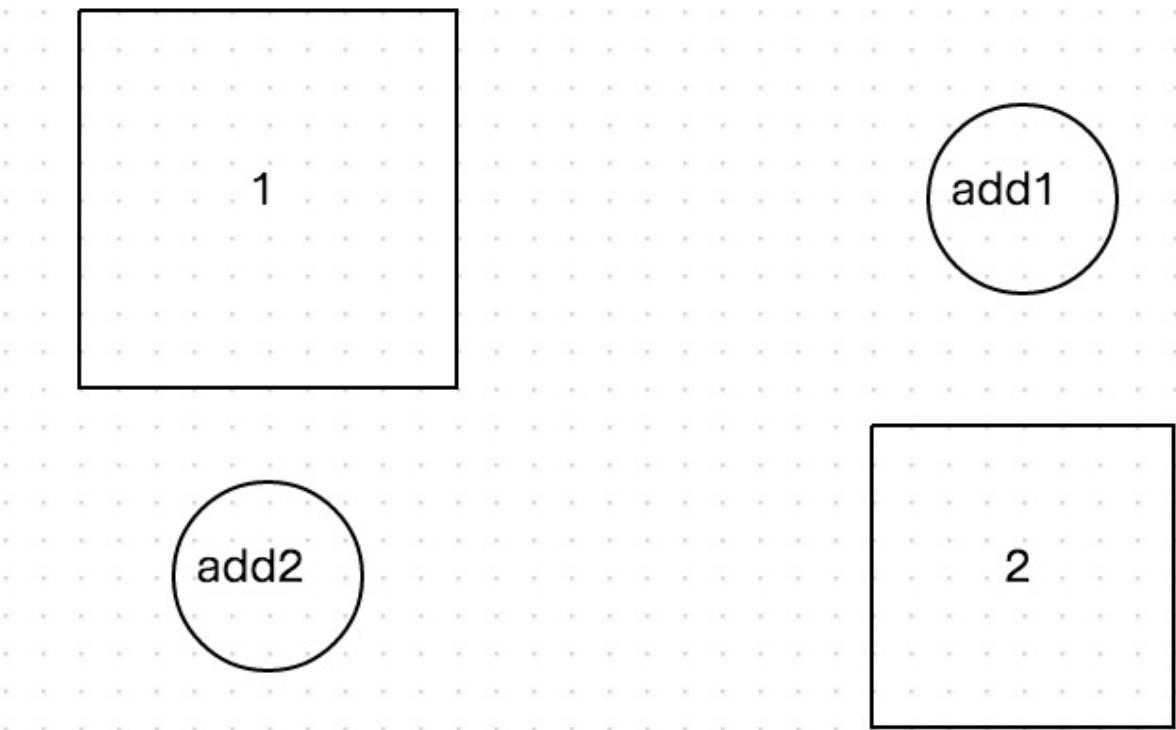
존재하는 그룹에 다른 도형을 집어넣는 예제입니다.

```
var canvas = new OG.Canvas('canvas', [1000, 800], 'white',
'url(resources/images/symbol/grid.gif');

var rectangleShape = canvas.drawShape([100, 100], new OG.RectangleShape('1'),
[100, 100]);
var circleShape = canvas.drawShape([300, 200], new OG.RectangleShape('2'), [80,
80]);

var group = canvas.group([rectangleShape,circleShape]);

var addShape1 = canvas.drawShape([300, 100], new OG.CircleShape('add1'), [50,
50]);
var addShape2 = canvas.drawShape([100, 200], new OG.CircleShape('add2'), [50,
50]);
canvas.addToGroup(group,[addShape1,addShape2]);
```



## clear

화면상의 모든 도형들을 삭제합니다.

```
canvas.clear();
```

## removeShape

주어진 도형을 캔버스에서 삭제합니다. 이때 도형과 연결된 연결선이 있다면 함께 삭제되고, 자식 요소가 있어도 함께 삭제됩니다.

```
var rectangleShape = canvas.drawShape([100, 100], new OG.RectangleShape('1'),  
[100, 100]);  
canvas.removeShape(rectangleShape);
```

## removeChild

주어진 도형의 하위요소만 삭제합니다.

```
var rectangleShape = canvas.drawShape([100, 100], new OG.RectangleShape('1'),  
[100, 100]);  
var circleShape = canvas.drawShape([300, 200], new OG.RectangleShape('2'), [80,  
80]);  
  
var group = canvas.group([rectangleShape,circleShape]);  
canvas.removeShape(group);
```

## getRoot

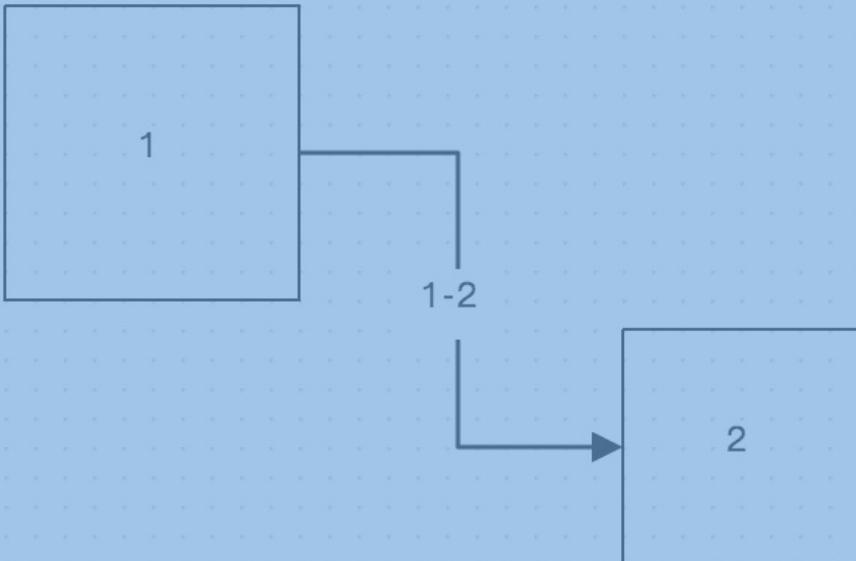
### getRootElement

오픈그래프의 캔버스 객체를 생성시키는 `var canvas = new OG.Canvas('canvas', [1000, 800]);` 를 실행할 때, 'canvas' 아이디를 가진 Div 엘리먼트를 인자값으로 주었습니다.

오픈그래프가 생성될 때는 이 'canvas' 아이디 하위 엘리먼트로 SVG 캔버스를 생성시키게 됩니다.

`getRootElement` 를 통해 SVG 캔버스 엘리먼트를 가져와서, 이 캔버스에 이벤트 바인딩 등의 커스텀 한 작업을 수행할 수 있습니다.

```
var rootElement = canvas.getRootElement();
```



**svg#0G\_1278 | 400 × 400**

## getRootGroup

계속해서, 'canvas' 하위의 SVG 앤리먼트한에, g 태그로 되어있는 엘리먼트가 존재하게 됩니다.

SVG 스펙에서는 g 태그를 그룹 엘리먼트라고 합니다.

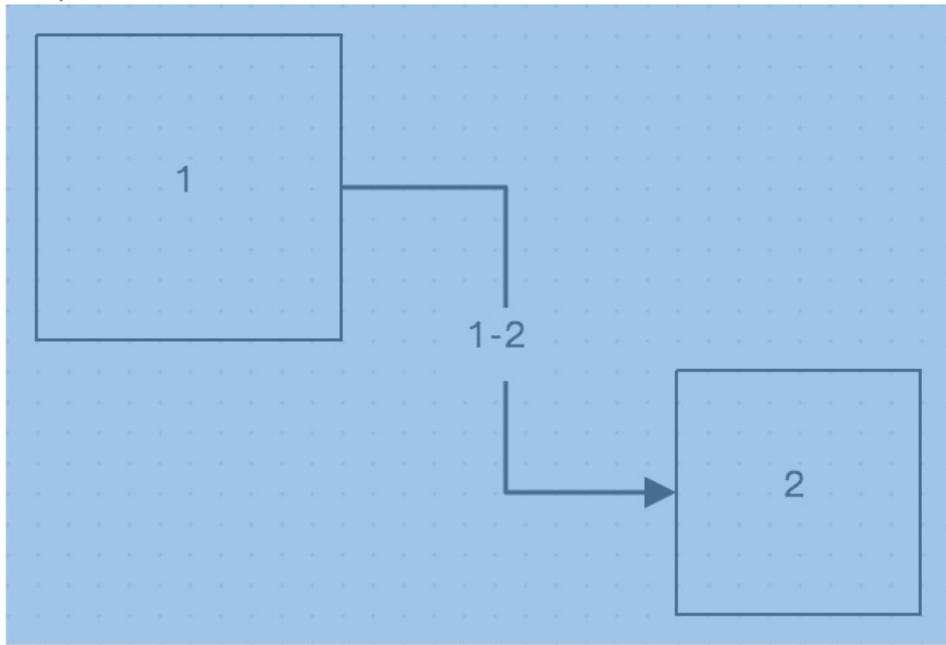
모든 오픈그래프 도형들은 g 태그를 가지고 있고, g 태그 안에 벡터 패스로 구성된 기하학 도형 또는 Image 요소들로 이루어져 있습니다.

이 중에서 가장 최상단의 g 태그를 RootGroup 이라 하고, 모든 오픈그래프의 도형들은 이 RootGroup 안에 존재하게 됩니다.

RootGroup 인 g 태그는 getRootGroup 메소드로 알 수 있습니다.

```
var rootGroup = canvas.getRootGroup();
```

g#0G\_1278\_0 | 290 × 190



## attribute

setAttr 와 getAttr 메소드로 도형에 속성값을 설정하고 가져올 수 있습니다.

```
var canvas = new OG.Canvas('canvas', [400, 400], 'white',
'url(resources/images/symbol/grid.gif');

var rectangleShape = canvas.drawShape([100, 100], new OG.RectangleShape('1'),
[100, 100]);
var id = canvas.getAttr(rectangleShape, 'id');

//오브젝트로 Attribute 설정
canvas.setAttr(rectangleShape, {
  cursor: 'none',
  fill: 'red'
});

var attr1 = canvas.getAttr(rectangleShape, 'cursor');
var attr2 = canvas.getAttr(rectangleShape, 'fill');

alert(id + ', ' + attr1 + ', ' + attr2);
```

## 주의

`setAttr` 은 Svg 스타일 요소가 아닌경우 적용되지 않습니다. (`cursor, fill, stroke` etc...) 따라서, 다른 종류의 어트리뷰트 요소를 세팅하고 싶으신 경우 Jquery 를 사용하시길 바랍니다.

ex) `$(rectangleShape).attr(attr, value);`

## front and back

다음 예제들은 도형을 앞으로 또는 뒤로 이동시키는 메소드입니다.

```
var circleShape = canvas.drawShape([101, 107], new OG.CircleShape(), [101, 102]);
var ellipseShape = canvas.drawShape([203, 405], new OG.EllipseShape('label'), [103, 53]);

//ID에 해당하는 Element 를 앞으로 한단계 이동한다.
canvas.bringForward(circleShape);

//ID에 해당하는 Element 를 뒤로 한단계 이동한다.
canvas.sendBackward(circleShape);

//ID에 해당하는 Element 를 최상단 레이어로 이동한다.
canvas.toFront(circleShape);

//ID에 해당하는 Element 를 최하단 레이어로 이동한다.
canvas.toBack(circleShape);
```

## canvas size

다음 예제들은 캔버스의 전체 사이즈에 관한 메소드입니다.

```

/**
 * 랜더러 캔버스의 사이즈(Width, Height)를 반환한다.
 *
 * @return {Number[]} Canvas Width, Height
 */
canvas.getCanvasSize();

/**
 * 랜더러 캔버스의 사이즈(Width, Height)를 변경한다.
 *
 * @param {Number[]} size Canvas Width, Height
 */
canvas.setCanvasSize([1000,500]);

/**
 * 랜더러 캔버스의 사이즈(Width, Height)를 실제 존재하는 Shape 의 영역에 맞게 변경한다.
 *
 * @param {Number[]} minSize Canvas 최소 Width, Height
 * @param {Boolean} fitScale 주어진 minSize 에 맞게 fit 여부(Default:false)
 */
canvas.fitCanvasSize([200,200],false);

```

## scale

다음 예제들은 캔버스의 Scale 에 관한 메소드 입니다.

```

/**
 * Scale 을 반환한다. (리얼 사이즈 : Scale = 1)
 *
 * @return {Number} 스케일값
 */
canvas.getScale();

/**
 * Scale 을 설정한다. (리얼 사이즈 : Scale = 1)
 *
 * @param {Number} scale 스케일값
 */
canvas.setScale(0.5);

```

## show and hide

다음 예제들은 도형의 숨김 / 보임 처리에 관한 메소드 입니다.

```
var circleShape = canvas.drawShape([101, 107], new OG.CircleShape(), [101, 102]);  
  
/**  
 * ID에 해당하는 Element 를 캔버스에서 show 한다.  
 *  
 * @param {Element|String} element Element 또는 ID  
 */  
canvas.show(circleShape);  
,  
  
/**  
 * ID에 해당하는 Element 를 캔버스에서 hide 한다.  
 *  
 * @param {Element|String} element Element 또는 ID  
 */  
canvas.hide(circleShape);
```

## append and insert

다음 예제들은 주어진 도형을 다른 도형의 자식으로 append 하거나, 다른 도형의 앞 또는 뒤에 insert 하는 예제입니다.

```

var parent = canvas.drawShape([100, 100], new OG.GroupShape(), [80, 80],
{stroke: 'blue'}, 'parentShape');
var element = canvas.drawShape([100, 100], new OG.CircleShape(), [50, 50],
{stroke: 'red'}, 'CircleShape');

/**
* Source Element 를 Target Element 아래에 append 한다.
*
* @param {Element|String} srcElement Element 또는 ID
* @param {Element|String} targetElement Element 또는 ID
* @return {Element} Source Element
*/
canvas.appendChild(element, parent);

/**
* Source Element 를 Target Element 이후에 insert 한다.
*
* @param {Element|String} srcElement Element 또는 ID
* @param {Element|String} targetElement Element 또는 ID
* @return {Element} Source Element
*/
canvas.insertAfter(element, parent);

/**
* Source Element 를 Target Element 이전에 insert 한다.
*
* @param {Element|String} srcElement Element 또는 ID
* @param {Element|String} targetElement Element 또는 ID
* @return {Element} Source Element
*/
canvas.insertBefore(element, parent);

```

## move

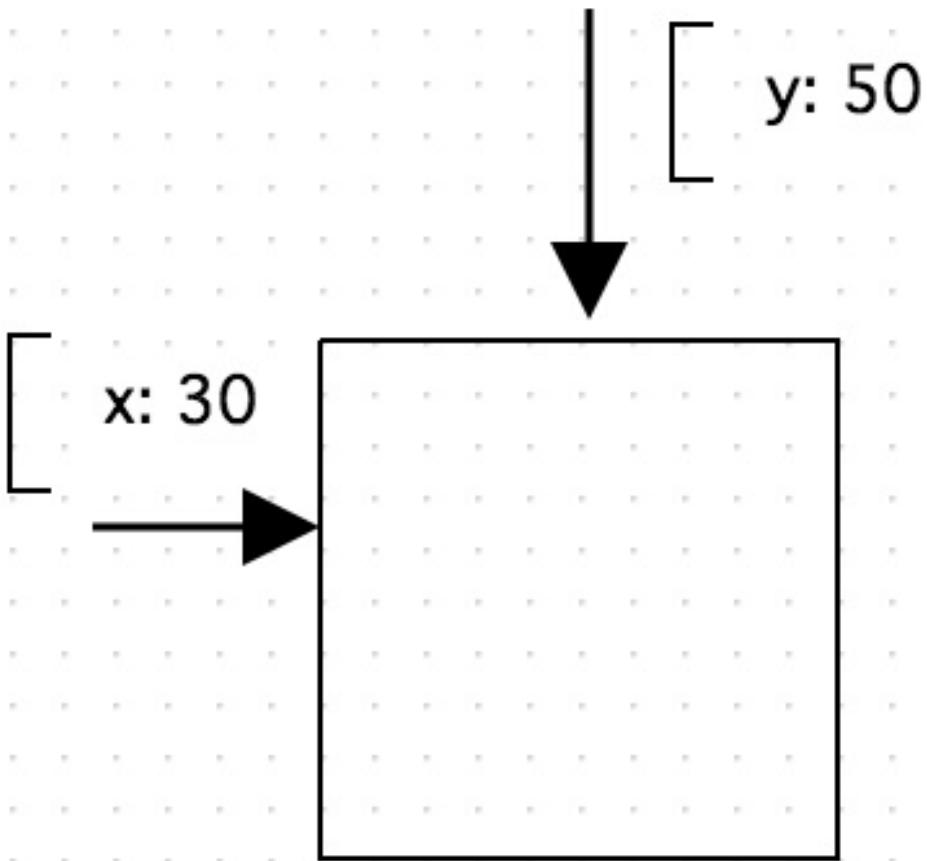
도형을 이동시킬 수 있는 방법은 두가지가 있습니다.

하나는 move 메소드를 통해 x,y 각각 이동할 증분을 설정하는 방법입니다. (- 값도 가능합니다.)

```

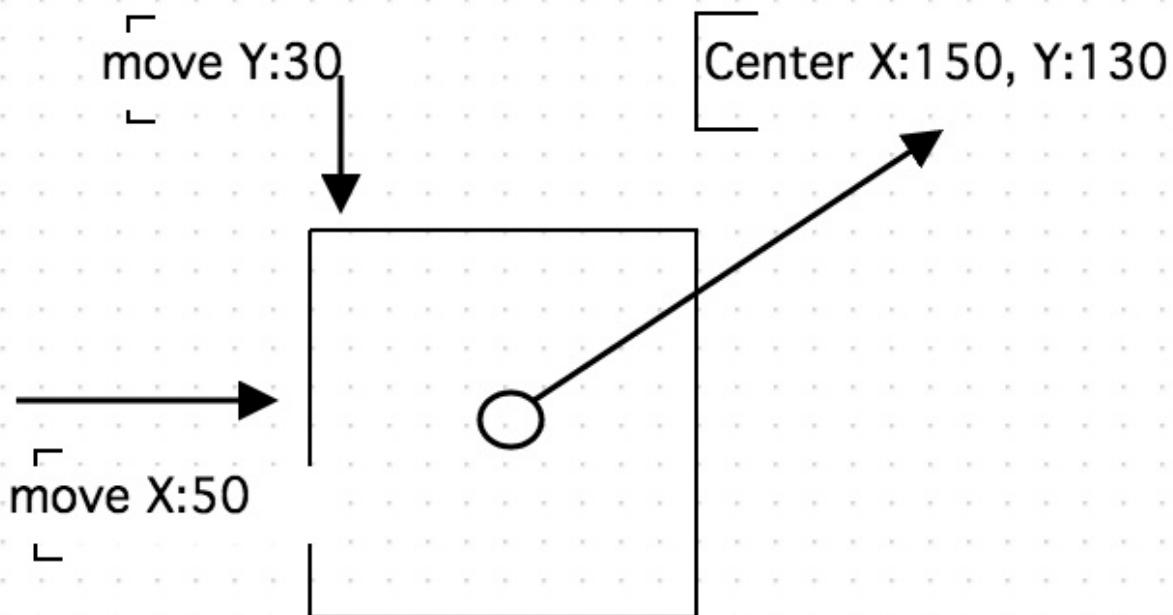
var rectShape = canvas.drawShape([100, 100], new OG.RectangleShape(), [100,
100]);
canvas.move(rectShape, [30, 50]);

```



또 다른 하나는 moveCentroid 메소드를 통해 중심 좌표를 설정하면, 도형이 중심좌표가 주어진 중심좌표와 일치하게 이동되게 됩니다.

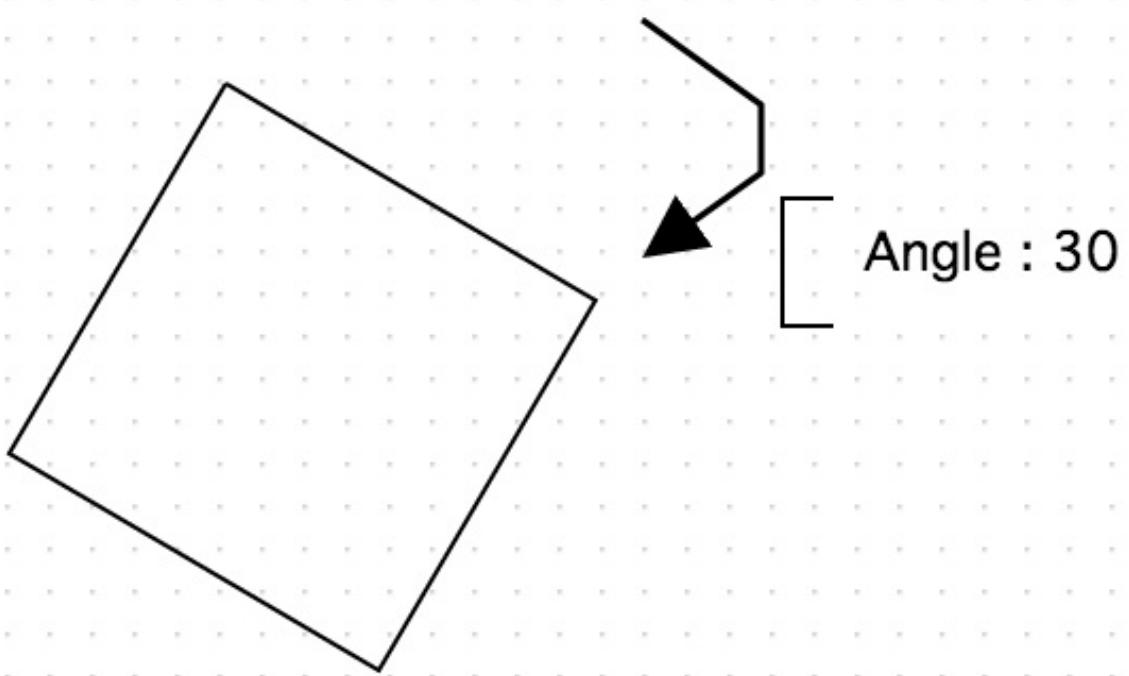
```
var rectShape = canvas.drawShape([100, 100], new OG.RectangleShape(), [100, 100]);  
canvas.moveCentroid(rectShape, [150, 130]);
```



## rotate

도형의 기울기에 관한 메소드입니다.

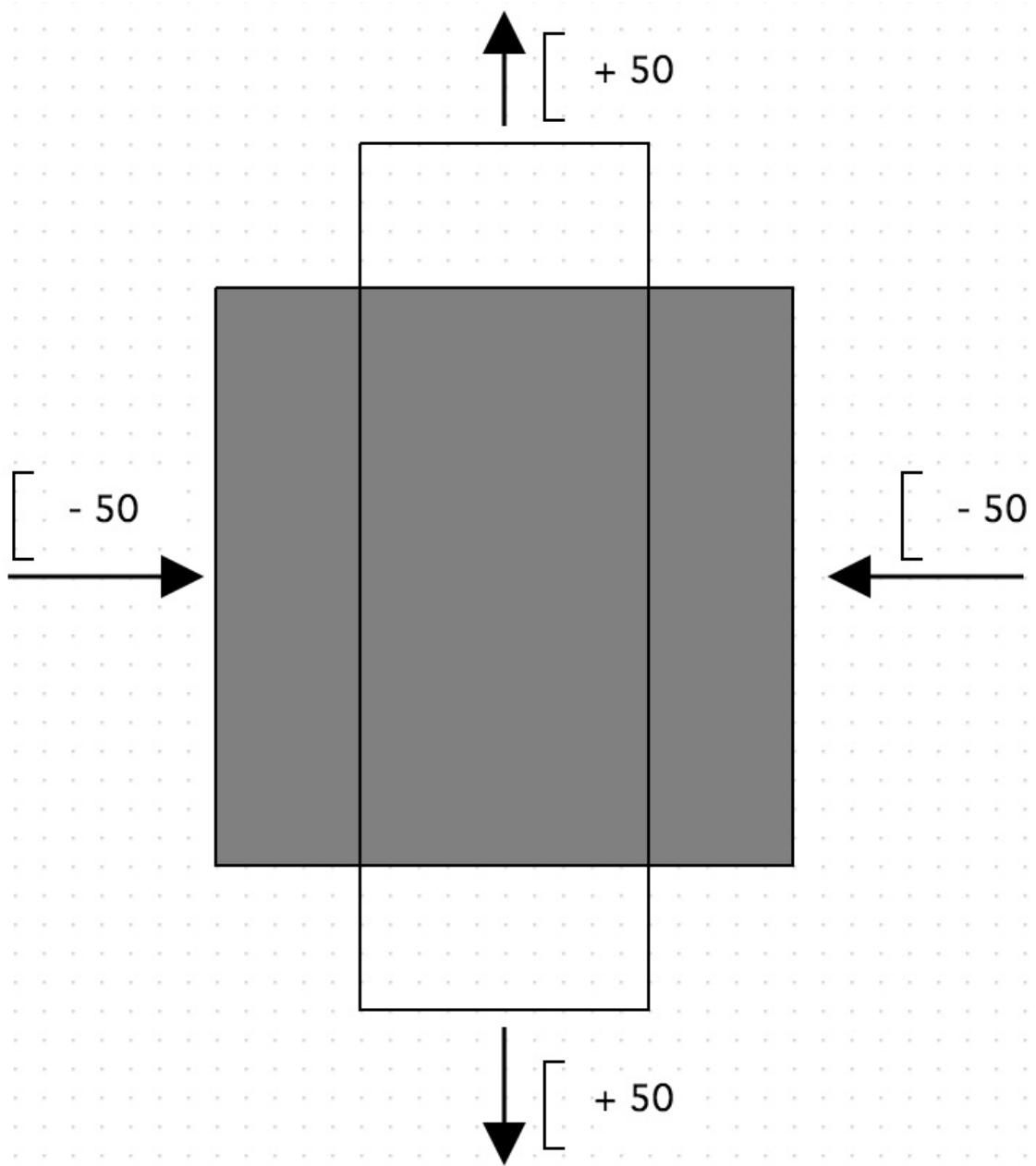
```
/**  
 * 중심 좌표를 기준으로 주어진 각도 만큼 회전한다.  
 *  
 * @param {Element|String} element Element 또는 ID  
 * @param {Number} angle 각도  
 * @return {Element} Element  
 */  
var rectShape = canvas.drawShape([100, 100], new OG.RectangleShape(), [100, 100]);  
canvas.rotate(rectShape, 30);
```



## resize

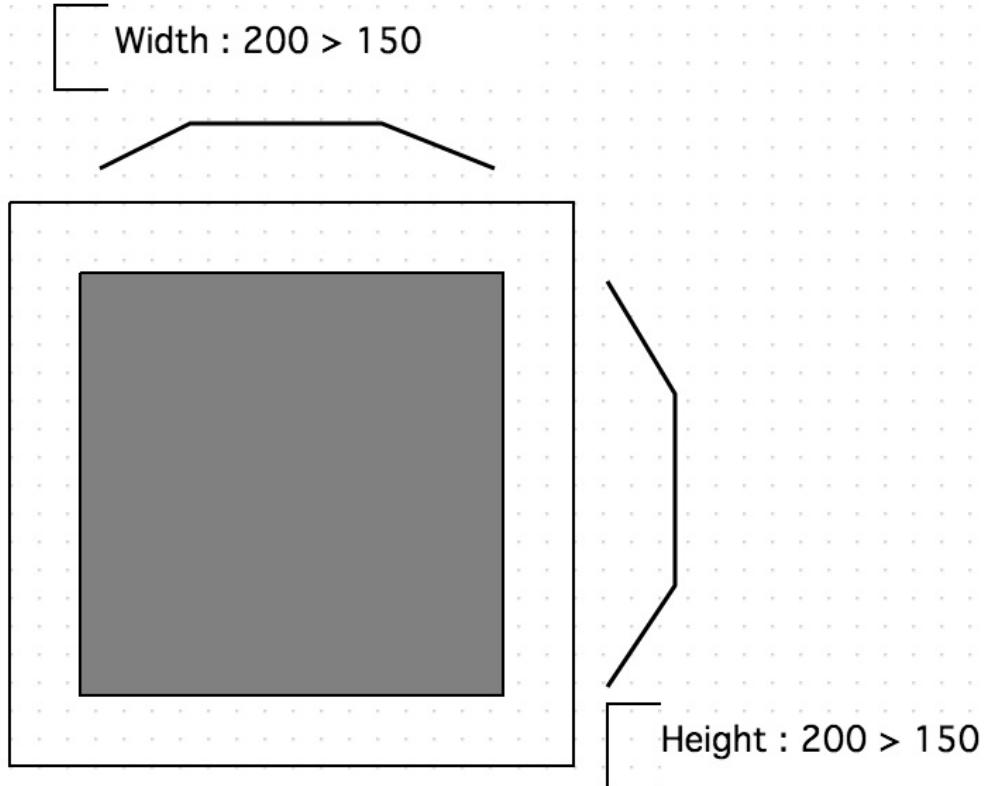
도형의 크기 조절에 관한 메소드입니다.

```
/**  
 * 상, 하, 좌, 우 외곽선을 이동한 만큼 리사이즈 한다.  
 *  
 * @param {Element|String} element Element 또는 ID  
 * @param {Number[]} offset [상, 하, 좌, 우] 각 방향으로 + 값  
 * @return {Element} Element  
 */  
  
var rectShape = canvas.drawShape([200, 200], new OG.RectangleShape(), [200, 200]);  
canvas.resize(rectShape, [50, 50, -50, -50]);
```



또 다른 방법으로는 `resizeBox` 메소드를 사용하여 중심좌표는 고정한 채, 가로와 세로 크기를 변경하는 방법입니다.

```
var rectShape = canvas.drawShape([200, 200], new OG.RectangleShape(), [200, 200]);
canvas.resizeBox(rectShape, [150, 150]);
```



## align

다음은 캔버스의 도형들의 align 상태에 관한 예제입니다.

align 관련 메소드들은 선택된 도형들 기준으로만 적용되며, 선택된 도형들 중 상,하,좌,우에 가장 가까운 도형 기준으로 선택된 도형들이 일괄 정렬됩니다.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ko" lang="ko">
<head>
    <title>BPMN Modeler Example</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
    <meta http-equiv="X-UA-Compatible" content="IE=EmulateIE8">

    <!-- jquery -->
    <script type="text/javascript" src=".//lib/jquery-1.11.1/jquery-
1.11.1.min.js"></script>

    <!-- jquery ui -->
    <script type="text/javascript" src=".//lib/jquery-ui-1.11.0.custom/jquery-
ui.min.js"></script>
    <link rel="stylesheet" type="text/css" href=".//lib/jquery-ui-
1.11.0.custom/jquery-ui.css"/>
```

```

<!-- jquery Context Menu -->
<link rel="stylesheet" type="text/css"
href="./lib/contextmenu/jquery.contextMenu.css"/>
<script type="text/javascript" src="./lib/contextmenu/jquery.contextMenu-
min.js"></script>

<!-- Opengraph -->
<script type="text/javascript" src="./lib/opengraph/OpenGraph-0.1.1-
SNAPSHOT.js"></script>

<script type="text/javascript">
$(document).ready(function () {
    var canvas = new OG.Canvas('canvas', [400, 400], 'white',
'url(resources/images/symbol/grid.gif'));

    var circleShape = canvas.drawShape([101, 107], new
OG.CircleShape(), [101, 102]);
    var ellipseShape = canvas.drawShape([203, 205], new
OG.EllipseShape('label'), [103, 53]);
    var edge = canvas.connect(circleShape, ellipseShape, null, '');

    canvas._HANDLER.selectShapes([circleShape,ellipseShape]);

    $('#alignLeft').click(function(){
        canvas.alignLeft();
    });

    $('#alignRight').click(function(){
        canvas.alignRight();
    });

    $('#alignTop').click(function(){
        canvas.alignTop();
    });

    $('#alignBottom').click(function(){
        canvas.alignBottom();
    });
});

</script>
</head>
<body>

<button id="alignLeft">alignLeft</button>
<button id="alignRight">alignRight</button>
<button id="alignTop">alignTop</button>
<button id="alignBottom">alignBottom</button>
<div id="canvas" style="cursor: default;"></div>
<div id="canvas_slider"></div>

```

```
</body>  
</html>
```

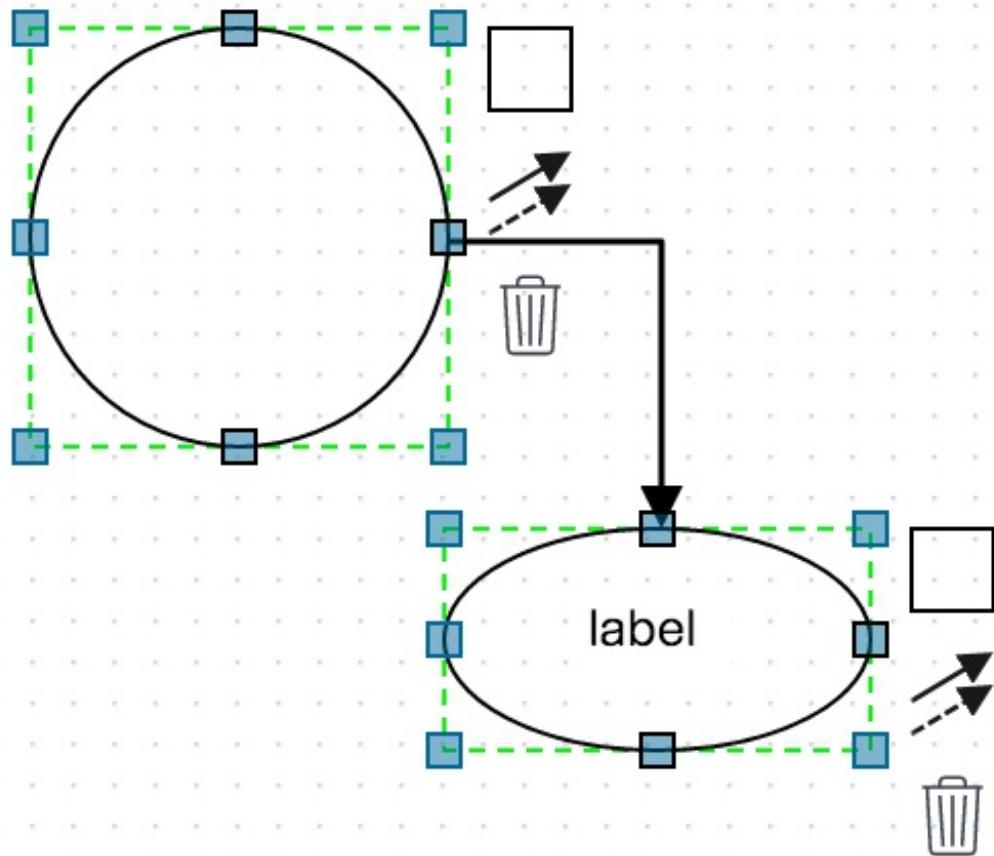
- 예제를 실행시킨 모습

**alignLeft**

**alignRight**

**alignTop**

**alignBottom**



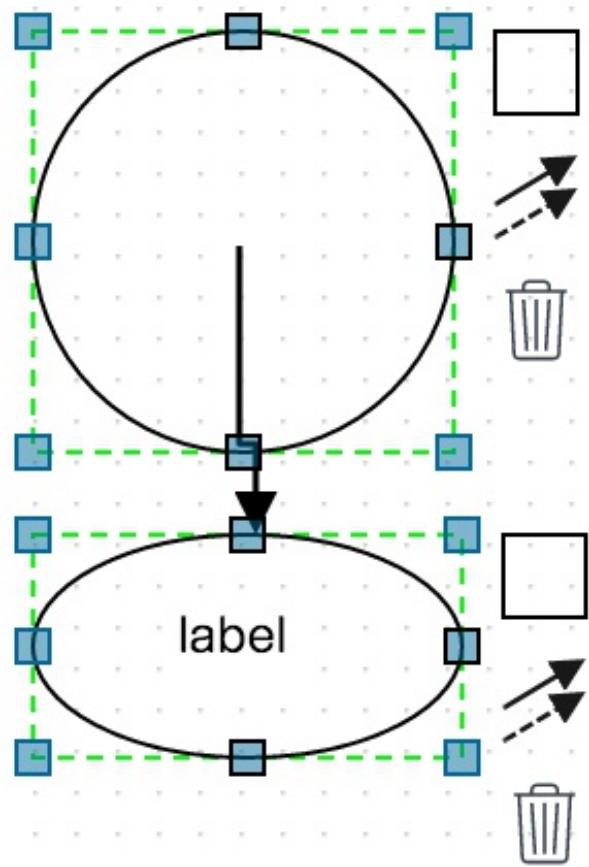
- alignLeft

**alignLeft**

**alignRight**

**alignTop**

**alignBottom**



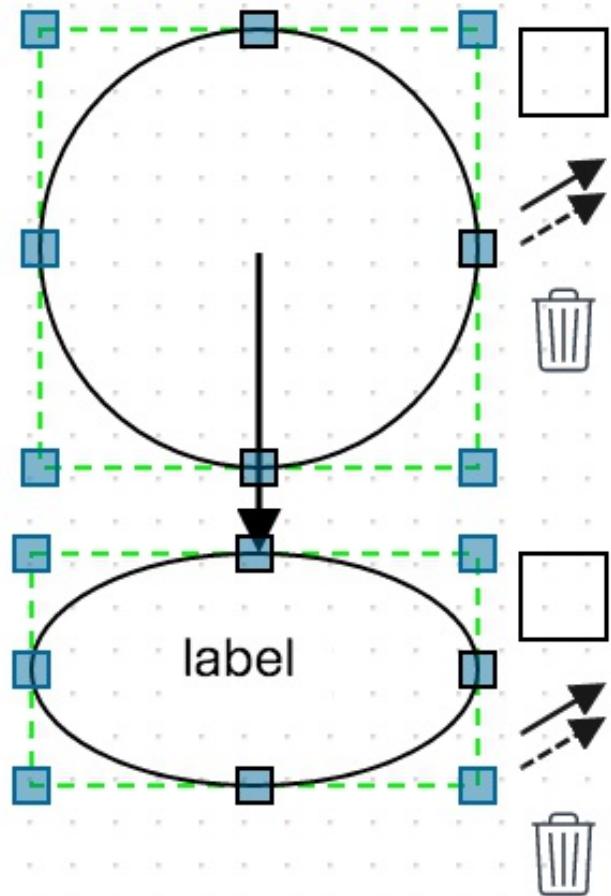
- `alignRight`

**alignLeft**

**alignRight**

**alignTop**

**alignBottom**



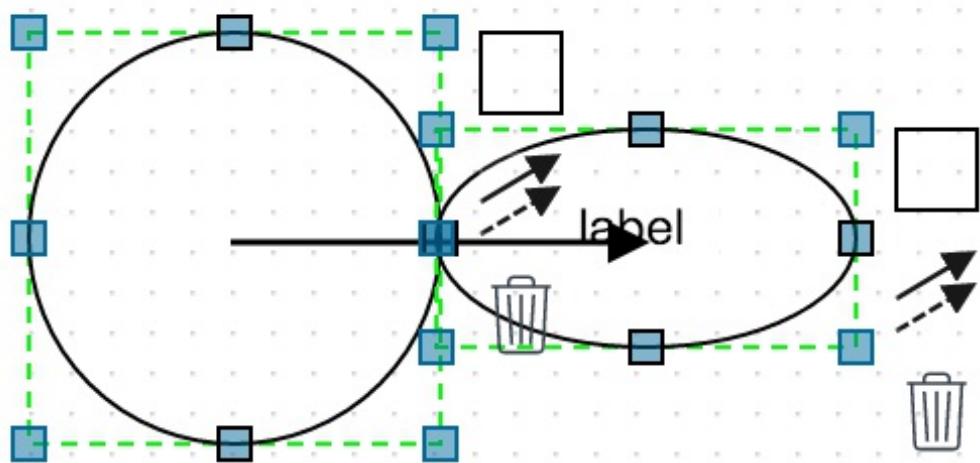
- alignTop

`alignLeft`

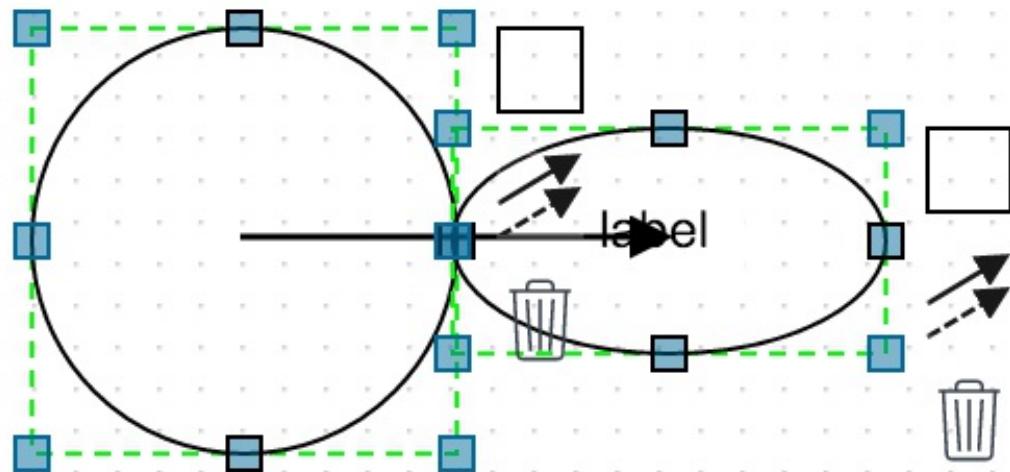
`alignRight`

`alignTop`

`alignBottom`



- `alignBottom`

[alignLeft](#)[alignRight](#)[alignTop](#)[alignBottom](#)

## undo && redo

사용자에 의해 편집된 오픈그래프의 상태를 undo , redo 합니다.

프로그램적으로 오픈그래프에 그려진 객체들은 undo , redo 되지 않습니다.

```
/**  
 * 캔버스 undo.  
 */  
canvas.undo();  
  
/**  
 * 캔버스 redo.  
 */  
canvas.redo();
```

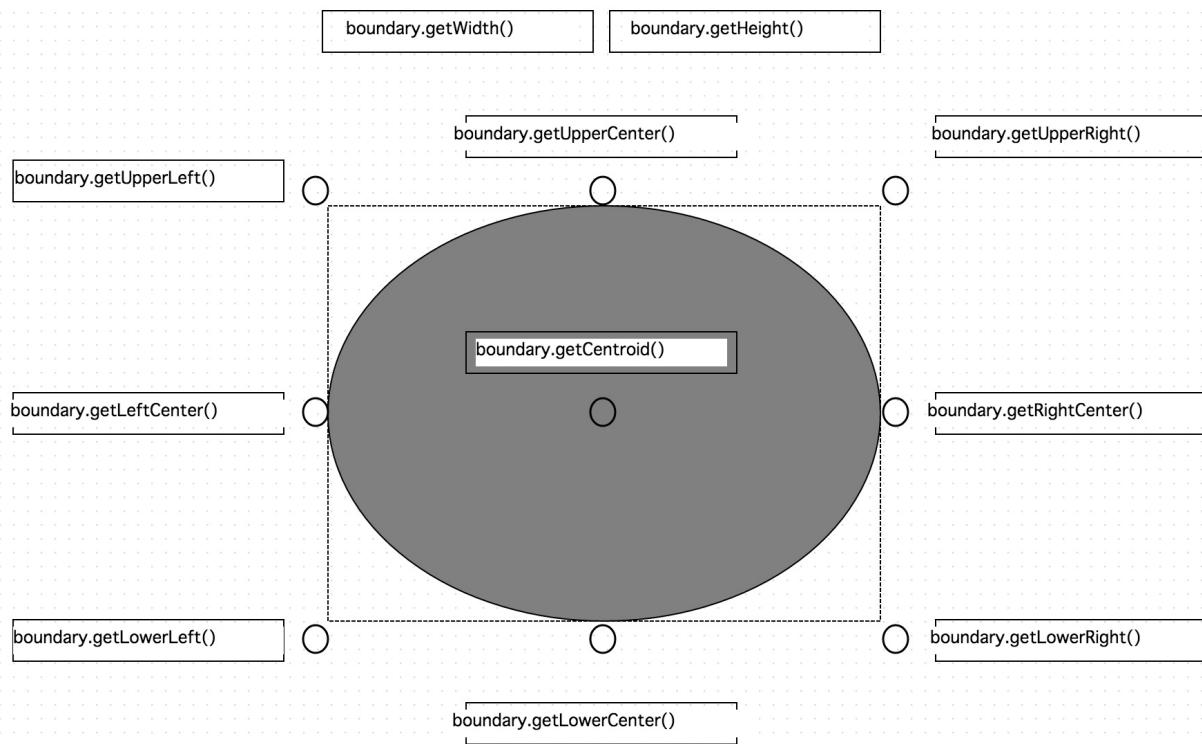
## getBoundary

getBoundary 메소드는 OG.geometry.Envelope 라는 도형의 바운더리 영역 객체를 리턴합니다.

바운더리 영역은 도형을 둘러싼 최소크기의 직사각형 영역을 의미합니다.

다음 예제는 getBoundary 메소드를 통해 얻은 OG.geometry.Envelope 객체에서 어떻게 값들을 가져올 수 있는지에 대한 예제입니다.

```
var element = canvas.drawShape([400, 300], new OG.EllipseShape(), [400, 300]);
var boundary = canvas.getBoundary(element);
console.log(boundary.getWidth());
console.log(boundary.getHeight());
console.log(boundary.getCentroid().x, boundary.getCentroid().y);
console.log(boundary.getUpperLeft().x,boundary.getUpperLeft().y);
console.log(boundary.getUpperCenter().x,boundary.getUpperCenter().y);
console.log(boundary.getUpperRight().x,boundary.getUpperRight().y);
console.log(boundary.getLeftCenter().x,boundary.getLeftCenter().y);
console.log(boundary.getRightCenter().x,boundary.getRightCenter().y);
console.log(boundary.getLowerLeft().x,boundary.getLowerLeft().y);
console.log(boundary.getLowerCenter().x,boundary.getLowerCenter().y);
console.log(boundary.getLowerRight().x,boundary.getLowerRight().y);
```



# Canvas Data

- [toXML && toJSON](#)
- [loadXML && loadJSON](#)
- [setCustomData && getCustomData](#)
- [setExtCustomData && getExtCustomData](#)

오픈그래프의 전체 캔버스의 데이터 입출력과 각 도형별 데이터 컨트롤에 대해 알아보겠습니다.

## toXML && toJSON

toXML && toJSON 은 캔버스에 그려진 모든 도형을 포함한 레이아웃 데이터를 xml 또는 Json 으로 반환합니다.

```
/**  
 *      Canvas 에 그려진 Shape 들을 OpenGraph XML 문자열로 export 한다.  
 *  
 * @return {String} XML 문자열  
 */  
canvas.toXML();  
  
/**  
 * Canvas 에 그려진 Shape 들을 OpenGraph JSON 객체로 export 한다.  
 *  
 * @return {Object} JSON 포맷의 Object  
 */  
canvas.toJSON();
```

## loadXML && loadJSON

toXML && toJSON 이 캔버스의 레이아웃 데이터를 반환하는 것이라면, loadXML && loadJSON 은 역으로 이 데이터를 캔버스에 드로잉 해 주는 메소드입니다.

이때 기존에 캔버스에 그려진 모든 도형들은 clear 됩니다.

```
/**  
 * OpenGraph XML 문자열로 부터 Shape 을 드로잉한다.  
 *  
 * @param {String| Element} xml XML 문자열 또는 DOM Element  
 * @return {Object} {width, height, x, y, x2, y2}  
 */  
canvas.loadXML(xml);  
  
/**  
 * JSON 객체로 부터 Shape 을 드로잉한다.  
 *  
 * @param {Object} json JSON 포맷의 Object  
 * @return {Object} {width, height, x, y, x2, y2}  
 */  
canvas.loadJSON(json)
```

## setCustomData && getCustomData

주어진 도형에 커스텀 데이터를 저장하거나 가져옵니다.

저장된 커스텀 데이터는 오픈그래프의 toJSON 메소드 또는 toXML 메소드로 전체 캔버스의 레이아웃을 출력할 때 함께 커스텀 데이터 역시 저장되며,  
loadJSON 또는 loadXML 메소드로 캔버스의 레이아웃을 가져올 때 커스텀 데이터 역시 복원됩니다.

```
/**  
 * 주어진 Shape 엘리먼트에 커스텀 데이터를 저장한다.  
 *  
 * @param {Element|String} shapeElement Shape DOM Element or ID  
 * @param {Object} data JSON 포맷의 Object  
 */  
canvas.setCustomData(element,data);  
/**  
 * 주어진 Shape 엘리먼트에 저장된 커스텀 데이터를 반환한다.  
 *  
 * @param {Element|String} shapeElement Shape DOM Element or ID  
 * @return {Object} JSON 포맷의 Object  
 */  
canvas.getCustomData(element,data);  
  
var element = canvas.drawShape([100, 100], new OG.CircleShape(), [50, 50],  
null, 'CircleShape');  
  
//커스텀 데이터를 저장한다.  
canvas.setCustomData(element, {index: 0, name: 'myShape'});  
  
//캔버스 레이아웃을 출력한다.  
var json = canvas.toJSON();  
  
//캔버스의 기존 요소를 삭제하고 레이아웃 데이터로 재구성한다.  
canvas.loadJSON(json);  
  
//주어진 아이디로 도형을 불러온다.  
var reloadedShape = canvas.getElementById('CircleShape');  
  
if (reloadedShape) {  
    //커스텀 데이터가 복원되어 불러와진다.  
    var customData = canvas.getCustomData(reloadedShape);  
    console.log(customData);  
}
```

## setExtCustomData && getExtCustomData

setCustomData && getCustomData 와 별개로 사용할 수 있는 확장 커스텀 데이터 메소드입니다.

마찬가지로 캔버스의 레이아웃을 가져오거나 불러올 때 복원됩니다.

```
/**  
 * 주어진 Shape 엘리먼트에 확장 커스텀 데이터를 저장한다.  
 *  
 * @param {Element|String} shapeElement Shape DOM Element or ID  
 * @param {Object} data JSON 포맷의 Object  
 */  
canvas.setExtCustomData(element,data);  
  
/**  
 * 주어진 Shape 엘리먼트에 저장된 확장 커스텀 데이터를 반환한다.  
 *  
 * @param {Element|String} shapeElement Shape DOM Element or ID  
 * @return {Object} JSON 포맷의 Object  
 */  
canvas.getExtCustomData(element,data);
```

# Canvas Selector

- [getElementById](#)
- [getElementsByType](#)
- [getElementsByShapeId](#)
- [getRelatedElementsFromEdge](#)
- [getParent](#)
- [getChilds](#)
- [getAllShapes](#)
- [getPrevEdges](#)
- [getNextEdges](#)
- [getPrevShapes](#)
- [getPrevShapeIds](#)
- [getNextShapes](#)
- [getNextShapeIds](#)

오픈그래프의 캔버스의 셀렉터 메소드들에 대해 알아보겠습니다.

## getElementById

ID로 도형을 반환합니다.

```
/**  
 * ID로 Node Element 를 반환한다.  
 *  
 * @param {String} id  
 * @return {Element} Element  
 */  
  
var rectShape = canvas.drawShape([100, 100], new OG.RectangleShape(), [100,  
100], null, 'RectShape');  
var selectShape = canvas.getElementById('RectShape');  
console.log(selectShape);
```

## getElementsByType

도형의 Type에 따라 도형들을 반환합니다.

```

/**
 * Shape 타입에 해당하는 Node Element 들을 반환한다.
 *
 * @param {String} shapeType Shape 타입(GEOM, HTML, IMAGE, EDGE, GROUP), Null 이면
 * 모든 타입
 * @param {String} excludeType 제외 할 타입
 * @return {Element[]} Element's Array
 */
var rectShape = canvas.drawShape([100, 100], new OG.RectangleShape(), [100,
100], null, 'RectShape');
var circleShape = canvas.drawShape([100, 100], new OG.CircleShape(), [50, 50],
null, 'CircleShape');
var selectShapes = canvas.getElementsByType('GEOM');
console.log(selectShapes);

```

## getElementsByShapeId

도형의 shape ID 에 해당하는 도형들을 반환합니다.

```

/**
 * Shape ID에 해당하는 Node Element 들을 반환한다.
 *
 * @param {String} shapeId Shape ID
 * @return {Element[]} Element's Array
 */
var rectShape = canvas.drawShape([100, 100], new OG.RectangleShape(), [100,
100], null, 'RectShape');
var circleShape = canvas.drawShape([100, 100], new OG.CircleShape(), [50, 50],
null, 'CircleShape');

var shapeId = $(rectShape).attr('_shape_id');//OG.shape.RectangleShape
var selectShapes = canvas.getElementsByShapeId(shapeId);
console.log(selectShapes);

```

## getRelatedElementsFromEdge

주어진 연결선과 연결된 도형들을 구합니다.

```

/**
 * Edge 엘리먼트와 연결된 fromShape, toShape 엘리먼트를 반환한다.
 *
 * @param {Element|String} edgeElement Element 또는 ID
 * @return {Object}
 */
var rectShape = canvas.drawShape([100, 100], new OG.RectangleShape(), [100, 100], null, 'RectShape');
var circleShape = canvas.drawShape([200, 200], new OG.CircleShape(), [50, 50], null, 'CircleShape');

var edge = canvas.connect(rectShape,circleShape);
var relatedElementsFromEdge = canvas.getRelatedElementsFromEdge(edge);
console.log(relatedElementsFromEdge.from); ==> rectShape
console.log(relatedElementsFromEdge.to); ==> circleShape

```

## getParent

```

/**
 * 부모 엘리먼트를 반환한다. 부모가 루트일때는 반환하지 않는다.
 *
 * @param {Element} Element 엘리먼트
 * @return {Element} Element 엘리먼트
 */
canvas.getParent(element);

```

## getChilds

```

/**
 * 그룹의 하위 엘리먼트를 반환한다.
 *
 * @param {Element} element 엘리먼트
 * @returns {Array} Elements
 */
canvas.getChilds(element);

```

## getAllShapes

```

/**
 * 캔버스의 모든 Shape 들을 리턴
 *
 * @return {Array} Elements
 */
canvas.getAllShapes();

```

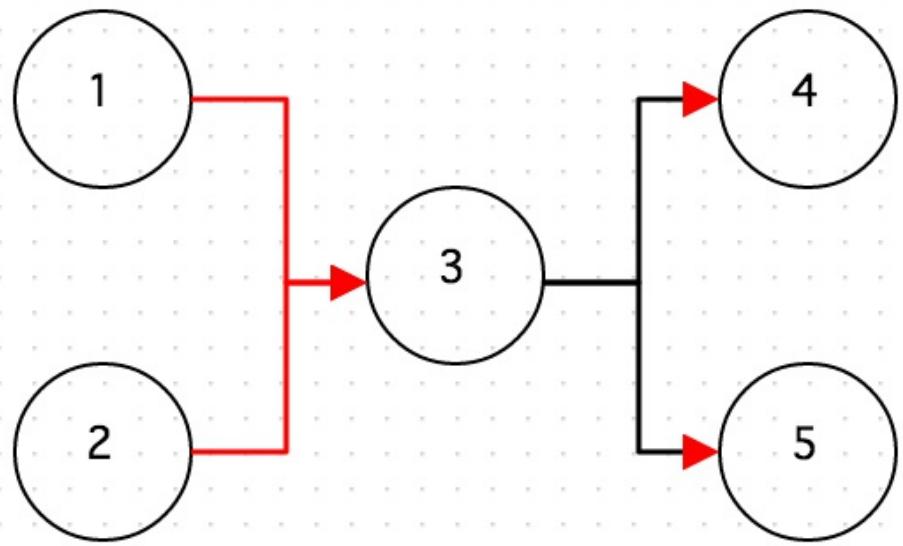
## getAllEdges

```
/**  
 * 캔버스의 모든 Edge를 리턴  
 *  
 * @return {Array} Edge Elements  
 */  
canvas.getAllEdges();
```

## getPrevEdges

주어진 도형과 연결된 이전 연결선들을 반환합니다.

```
/**  
 * 연결된 이전 Edge Element 들을 반환한다.  
 *  
 * @param {Element|String} element Element 또는 ID  
 * @return {Element[]} Previous Element's Array  
 */  
  
var circleShape1 = canvas.drawShape([100, 100], new OG.CircleShape('1'), [50, 50], null);  
var circleShape2 = canvas.drawShape([100, 200], new OG.CircleShape('2'), [50, 50], null);  
var circleShape3 = canvas.drawShape([200, 150], new OG.CircleShape('3'), [50, 50], null);  
var circleShape4 = canvas.drawShape([300, 100], new OG.CircleShape('4'), [50, 50], null);  
var circleShape5 = canvas.drawShape([300, 200], new OG.CircleShape('5'), [50, 50], null);  
  
canvas.connect(circleShape1, circleShape3);  
canvas.connect(circleShape2, circleShape3);  
canvas.connect(circleShape3, circleShape4);  
canvas.connect(circleShape3, circleShape5);  
  
var prevEdges = canvas.getPrevEdges(circleShape3);  
for (var i = 0, leni = prevEdges.length; i < leni; i++) {  
    canvas.setShapeStyle(prevEdges[i], {stroke: 'red'});  
}
```



## getNextEdges

주어진 도형과 연결된 Next 연결선들을 반환합니다.

```

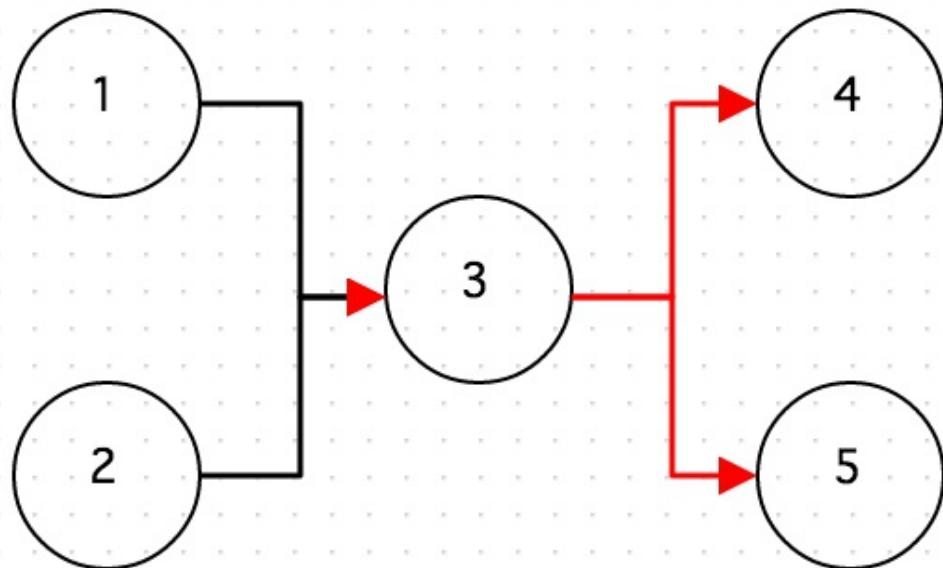
/**
 * 연결된 이후 Edge Element 들을 반환한다.
 *
 * @param {Element|String} element Element 또는 ID
 * @return {Element[]} Previous Element's Array
 */

var circleShape1 = canvas.drawShape([100, 100], new OG.CircleShape('1'), [50, 50], null);
var circleShape2 = canvas.drawShape([100, 200], new OG.CircleShape('2'), [50, 50], null);
var circleShape3 = canvas.drawShape([200, 150], new OG.CircleShape('3'), [50, 50], null);
var circleShape4 = canvas.drawShape([300, 100], new OG.CircleShape('4'), [50, 50], null);
var circleShape5 = canvas.drawShape([300, 200], new OG.CircleShape('5'), [50, 50], null);

canvas.connect(circleShape1, circleShape3);
canvas.connect(circleShape2, circleShape3);
canvas.connect(circleShape3, circleShape4);
canvas.connect(circleShape3, circleShape5);

var nextEdges = canvas.getNextEdges(circleShape3);
for (var i = 0, leni = nextEdges.length; i < leni; i++) {
    canvas.setShapeStyle(nextEdges[i], {stroke: 'red'});
}

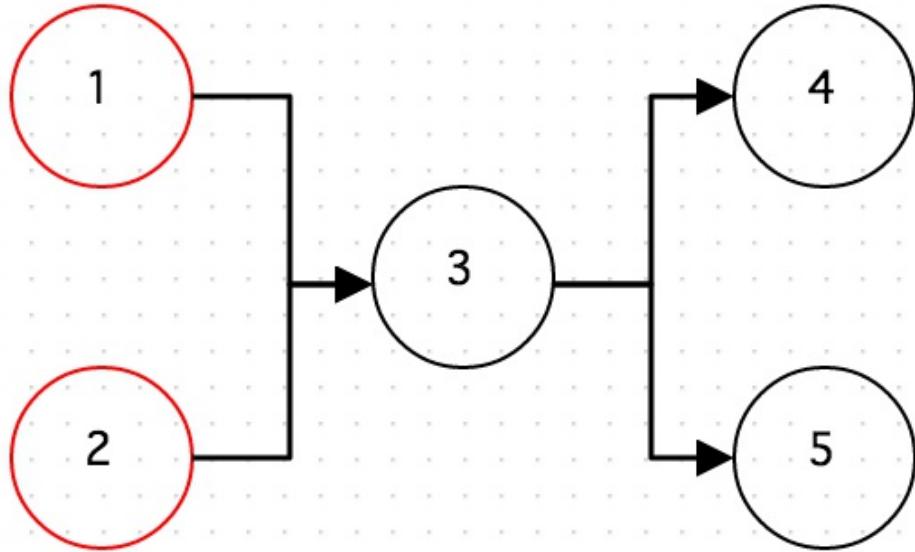
```



## getPrevShapes

주어진 도형과 연결된 이전 도형들을 반환합니다.

```
/**  
 * 연결된 이전 노드 Element 들을 반환한다.  
 *  
 * @param {Element|String} element Element 또는 ID  
 * @return {Element[]} Previous Element's Array  
 */  
  
var circleShape1 = canvas.drawShape([100, 100], new OG.CircleShape('1'), [50, 50], null);  
var circleShape2 = canvas.drawShape([100, 200], new OG.CircleShape('2'), [50, 50], null);  
var circleShape3 = canvas.drawShape([200, 150], new OG.CircleShape('3'), [50, 50], null);  
var circleShape4 = canvas.drawShape([300, 100], new OG.CircleShape('4'), [50, 50], null);  
var circleShape5 = canvas.drawShape([300, 200], new OG.CircleShape('5'), [50, 50], null);  
  
canvas.connect(circleShape1, circleShape3);  
canvas.connect(circleShape2, circleShape3);  
canvas.connect(circleShape3, circleShape4);  
canvas.connect(circleShape3, circleShape5);  
  
var prevShapes = canvas.getPrevShapes(circleShape3);  
for (var i = 0, leni = prevShapes.length; i < leni; i++) {  
    canvas.setShapeStyle(prevShapes[i], {stroke: 'red'});  
}
```



## getPrevShapeIds

주어진 도형과 연결된 이전 도형들의 아이디들을 반환합니다.

```

/**
 * 연결된 이전 노드 Element ID들을 반환한다.
 *
 * @param {Element|String} element Element 또는 ID
 * @return {String[]} Previous Element Id's Array
 */
canvas.getPrevShapeIds(element);
  
```

## getNextShapes

주어진 도형과 연결된 Next 도형들을 반환합니다.

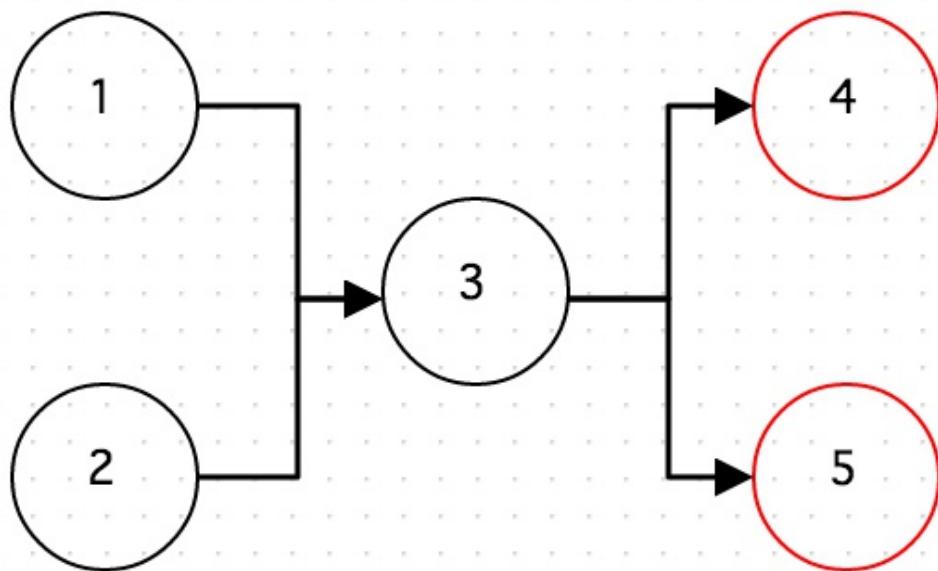
```

/**
 * 연결된 이후 노드 Element 들을 반환한다.
 *
 * @param {Element|String} element Element 또는 ID
 * @return {Element[]} Previous Element's Array
 */
var circleShape1 = canvas.drawShape([100, 100], new OG.CircleShape('1'), [50, 50], null);
var circleShape2 = canvas.drawShape([100, 200], new OG.CircleShape('2'), [50, 50], null);
var circleShape3 = canvas.drawShape([200, 150], new OG.CircleShape('3'), [50, 50], null);
var circleShape4 = canvas.drawShape([300, 100], new OG.CircleShape('4'), [50, 50], null);
var circleShape5 = canvas.drawShape([300, 200], new OG.CircleShape('5'), [50, 50], null);

canvas.connect(circleShape1, circleShape3);
canvas.connect(circleShape2, circleShape3);
canvas.connect(circleShape3, circleShape4);
canvas.connect(circleShape3, circleShape5);

var nextShapes = canvas.getNextShapes(circleShape3);
for (var i = 0, leni = nextShapes.length; i < leni; i++) {
    canvas.setShapeStyle(nextShapes[i],{stroke: 'red'});
}

```



## getNextShapeIds

주어진 도형과 연결된 Next 도형들의 아이디들을 반환합니다.

```
**  
* 연결된 이후 노드 Element ID들을 반환한다.  
*  
* @param {Element|String} element Element 또는 ID  
* @return {String[]} Previous Element Id's Array  
*/  
canvas.getNextShapeIds(element);
```

# Event

- [onDrawShape && onRedrawShape](#)
- [onUndo && onRedo](#)
- [onDivideLane](#)
- [onDrawLabel](#)
- [onLabelChanged](#)
- [onBeforeLabelChange](#)
- [onRemoveShape && onBeforeRemoveShape](#)
- [onRotateShape](#)
- [onMoveShape](#)
- [onResizeShape](#)
- [onBeforeConnectShape](#)
- [onConnectShape](#)
- [onDisconnectShape](#)
- [onGroup](#)
- [onUnGroup](#)

오픈그래프의 캔버스가 제공하는 이벤트에 대해 알아보도록 합니다.

## onDrawShape && onRedrawShape

onDrawShape && onRedrawShape 는 도형이 캔버스에 draw 되었을때의 이벤트 리스너입니다.

onDrawShape 는 canvas.drawShape 를 통해 draw 되었을 때의 리스너이며,  
onRedrawShape 는 move 혹은 resize, 또는 그룹 도형 이동시 내부 도형의 이동 등 기존의 그려진 도형이 사용자에 의해 다시 draw 되었을 경우 리스너입니다.

```

/**
 * Shape 이 처음 Draw 되었을 때의 이벤트 리스너
 *
 * @param {Function} callbackFunc 콜백함수(event, shapeElement)
 */
canvas.onDrawShape(function(event,element){
    console.log('onDrawShape', element);
});

/**
 * Shape 이 Redraw 되었을 때의 이벤트 리스너
 *
 * @param {Function} callbackFunc 콜백함수(event, shapeElement)
 */
canvas.onRedrawShape(function(event,element){
    console.log('onRedrawShape', element);
});

//onDrawShape 리스너가 실행됨.
var element = canvas.drawShape([100, 100], new OG.CircleShape(), [50, 50],
null, 'CircleShape1');

//onRedrawShape 리스너가 실행됨.
canvas.move(element,[-30,0]);

//onRedrawShape 리스너가 실행됨.
canvas.resize(element,[5,5,5,5]);

```

## onUndo && onRedo

캔버스가 undo 또는 redo 되었을때의 리스너 입니다.

```

/**
 * Undo 되었을때의 이벤트 리스너
 *
 * @param {Function} callbackFunc 콜백함수(event)
 */
canvas.onUndo(function(event){
    console.log('onUndo');
});

/**
 * Redo 되었을때의 이벤트 리스너
 *
 * @param {Function} callbackFunc 콜백함수(event)
 */
canvas.onredo(function(event){
    console.log('onRedo');
});

```

## onDivideLane

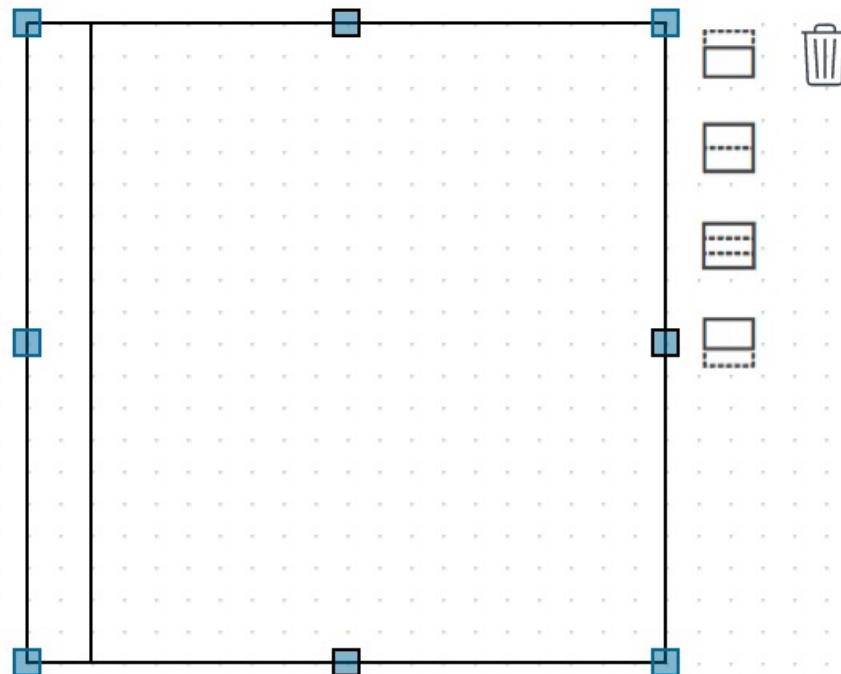
오픈그래프에는 Lane 도형이 기본으로 제공되는데, 이 Lane 도형은 분기할 수 있는 ui 인터페이스를 제공합니다.

사용자가 gui 상에서 분기 실행을 하였을 경우 일어나는 이벤트입니다.

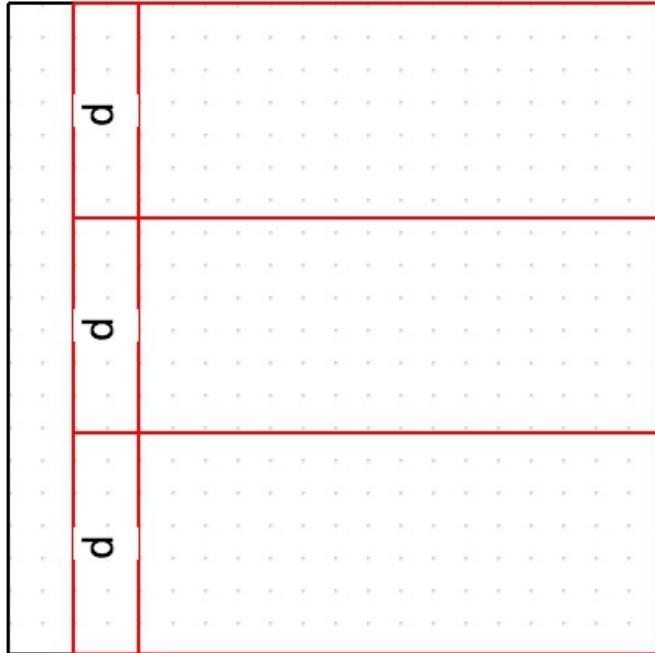
분기하여 새로 생성된 Lane 도형마다 이벤트를 발생시킵니다.

```
canvas.onDivideLane(function (event, dividedLane) {  
    canvas.drawLabel(dividedLane, 'd');  
    canvas.setShapeStyle(dividedLane, {stroke: 'red'});  
});  
  
var laneElement = canvas.drawShape([200, 200], new OG.HorizontalLaneShape(),  
[200, 200]);
```

- 분기 되기 전



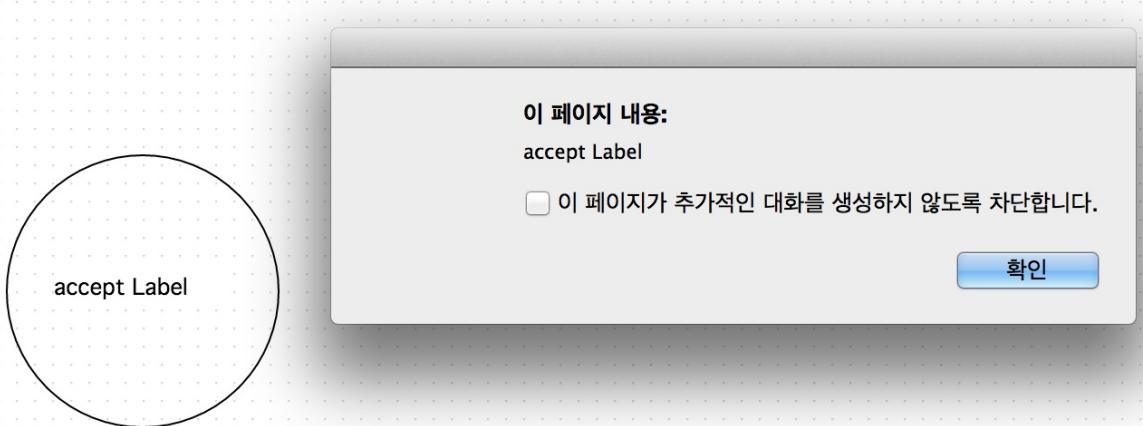
- 분기 된 후, 이벤트에 의해 stroke 바 바뀌었다.



## onDrawLabel

라벨이 Draw 되었을 때의 이벤트 리스너입니다.

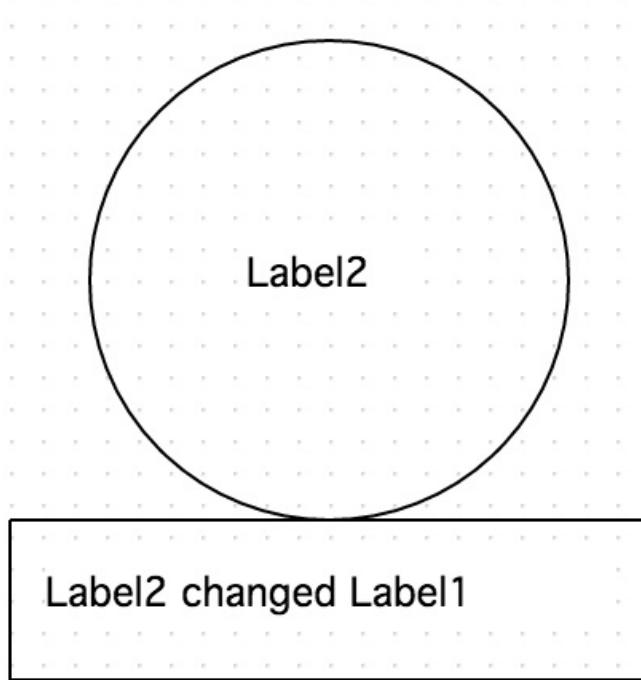
```
canvas.onDrawLabel(function (event, shapeElement, labelText) {  
    console.log(shapeElement, labelText);  
    setTimeout(function(){  
        alert(labelText);  
    }, 1000);  
});  
  
var circleShape = canvas.drawShape([200, 200], new OG.CircleShape(), [150,  
150]);  
canvas.drawLabel(circleShape, 'accept Label');
```



## onLabelChanged

사용자가 UI 상에서 라벨을 바꾸거나, 한번 그려진 라벨을 프로그램적으로 redraw 할 때의 이벤트 리스너입니다.

```
/**  
 * 라벨이 Change 되었을 때의 이벤트 리스너  
 *  
 * @param {Function} callbackFunc 콜백함수(event, shapeElement, afterText,  
beforeText)  
*/  
canvas.onLabelChanged(function(event, shapeElement, afterText, beforeText){  
    console.log(shapeElement, afterText, beforeText);  
    canvas.drawShape([200, 300], new OG.RectangleShape(beforeText + ' changed ' +  
+ afterText), [200, 50]);  
});  
  
var circleShape = canvas.drawShape([200, 200], new OG.CircleShape(), [150, 150]);  
  
canvas.drawLabel(circleShape, 'Label1');  
  
canvas.drawLabel(circleShape, 'Label2');
```



## onBeforeLabelChange

사용자가 UI 상에서 라벨을 바꾸거나, 한번 그려진 라벨을 프로그램적으로 redraw 할 때 라벨을 변경하기 전 발생되는 리스너입니다.

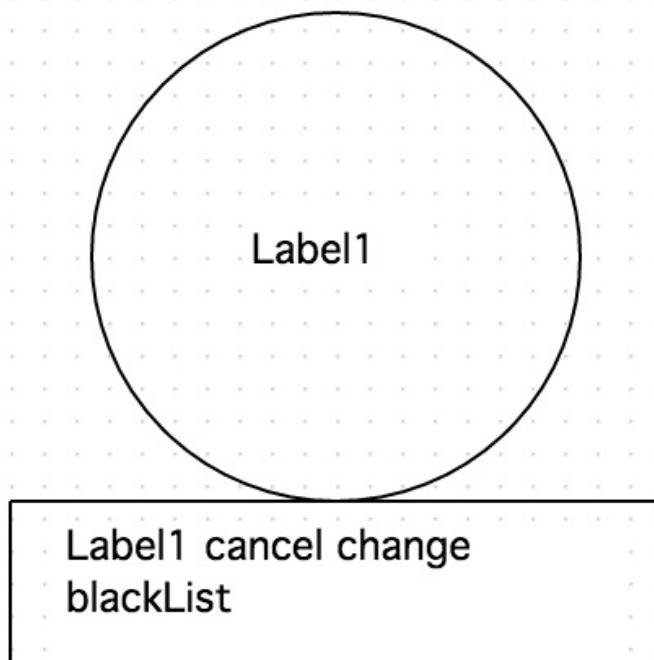
이 리스너에서 return 을 false 로 주게 되면 라벨이 변경되지 않습니다.

```

canvas.onBeforeLabelChange(function(event, shapeElement, afterText, beforeText)
{
    if(afterText == 'blackList'){
        console.log(shapeElement, afterText, beforeText);
        canvas.drawShape([200, 300], new OG.RectangleShape(beforeText + 'cancel change ' + afterText), [200, 50]);
        return false;
    }
});

var circleShape = canvas.drawShape([200, 200], new OG.CircleShape(), [150, 150]);
canvas.drawLabel(circleShape, 'Label1');
canvas.drawLabel(circleShape, 'blackList');

```



## onRemoveShape && onBeforeRemoveShape

사용자가 UI 상에서 도형을 삭제하거나, 프로그램적으로 삭제할 경우 onRemoveShape 리스너가 호출됩니다.

삭제하기 전 발생되는 리스너는 onBeforeRemoveShape입니다. 리턴 값을 false로 줄 경우 삭제되지 않습니다.

```
/**  
 * Shape 이 Remove 될 때의 이벤트 리스너  
 *  
 * @param {Function} callbackFunc 콜백함수(event, shapeElement)  
 */  
canvas.onRemoveShape(function(event, shapeElement){  
  
});  
  
canvas.onBeforeRemoveShape(function(event, shapeElement){  
  
});
```

## onRotateShape

도형이 Rotate 되었을 경우 이벤트 리스너입니다.

```
/**  
 * Shape 이 Rotate 될 때의 이벤트 리스너  
 *  
 * @param {Function} callbackFunc 콜백함수(event, element, angle)  
 */  
canvas.onRotateShape(function(event, element, angle){  
  
});
```

## onMoveShape

도형이 이동되었을 경우 이벤트 리스너입니다.

다음은 리스너를 통해 이동되었던 도형을 이동되었던 x,y 값 만큼 역으로 이동시킨 도형을 새로 그리는 예제입니다.

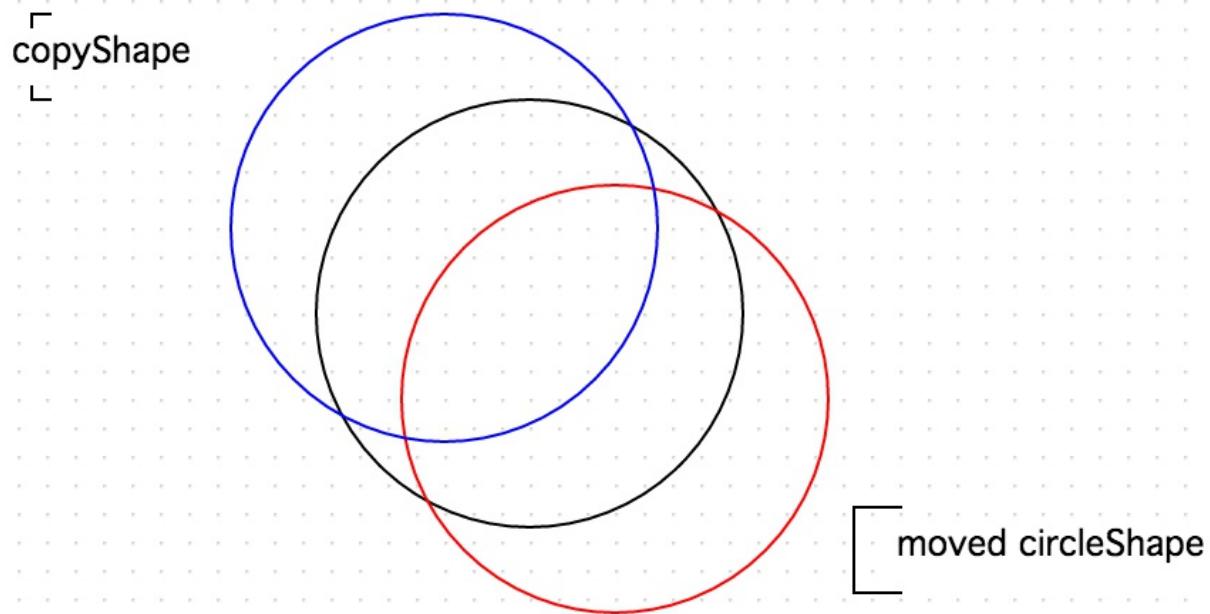
```

/**
 * Shape 이 Move 되었을 때의 이벤트 리스너
 *
 * @param {Function} callbackFunc 콜백함수(event, shapeElement, offset)
 */
var originalShape = canvas.drawShape([200, 200], new OG.CircleShape(), [150, 150]);

canvas.onMoveShape(function(event, shapeElement, offset){
    var boundary = canvas.getBoundary(shapeElement);
    var copyShape = canvas.drawShape(
        [boundary.getCentroid().x - (offset[0] * 2),
        boundary.getCentroid().y - (offset[1] * 2)],
        new OG.CircleShape(),
        [150, 150]);
    canvas.setShapeStyle(copyShape, {stroke: 'blue'});
});

var circleShape = canvas.drawShape([200, 200], new OG.CircleShape(), [150, 150]);
canvas.move(circleShape, [30, 30]);
canvas.setShapeStyle(circleShape, {stroke: 'red'});

```



## onResizeShape

도형이 리사이즈 되었을 경우의 이벤트 리스너 입니다.

콜백 값의 offset 은 상, 하, 좌, 우 각 방향으로 이동된 + 값을 리턴합니다.

```

/**
 * Shape 이 Resize 되었을 때의 이벤트 리스너
 *
 * @param {Function} callbackFunc 콜백함수(event, shapeElement, offset)
 */
canvas.onResizeShape(function (event, shapeElement, offset) {
});

```

## onBeforeConnectShape

도형이 연결되기 전 리스너이며, false 리턴을 주게 되면 연결이 취소됩니다.

```

/**
 * Shape 이 Connect 되기전 이벤트 리스너
 *
 * @param {Function} callbackFunc 콜백함수(event, edgeElement, fromElement,
toElement)
*/
canvas.onBeforeConnectShape(function(event, edgeElement, fromElement,
toElement){
});

```

## onConnectShape

```

/**
 * Shape 이 Connect 되었을 때의 이벤트 리스너
 *
 * @param {Function} callbackFunc 콜백함수(event, edgeElement, fromElement,
toElement)
*/
canvas.onConnectShape(function(event, edgeElement, fromElement, toElement){
});

```

## onDisconnectShape

```

/**
 * Shape 이 Disconnect 되었을 때의 이벤트 리스너
 *
 * @param {Function} callbackFunc 콜백함수(event, edgeElement, fromElement,
toElement)
*/
canvas.onDisconnectShape(function(event, edgeElement, fromElement, toElement){
});

```

## onGroup

```
/**  
 * Shape 이 Grouping 되었을 때의 이벤트 리스너  
 *  
 * @param {Function} callbackFunc 콜백함수(event, groupElement)  
 */  
canvas.onGroup(function(event, groupElement){  
  
});
```

## onUnGroup

```
/**  
 * Shape 이 UnGrouping 되었을 때의 이벤트 리스너  
 *  
 * @param {Function} callbackFunc 콜백함수(event, ungroupedElements)  
 */  
canvas.onUnGroup(function(event, ungroupedElements){  
  
});
```

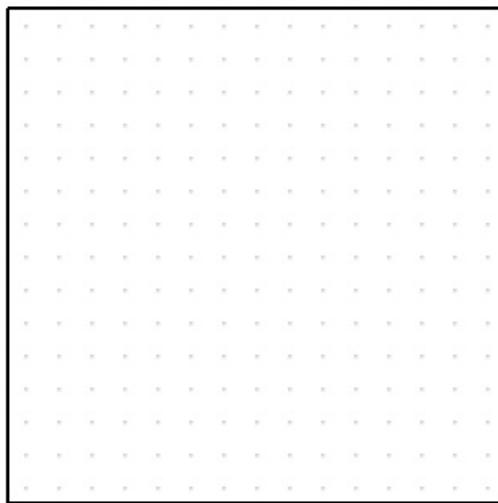
# Element

- [Structure](#)
- [Event binding](#)

## Structure

오픈그래프의 도형을 하나 그려보고, 도형을 콘솔로 출력하여 봅니다.

```
var canvas = new OG.Canvas('canvas', [1000, 800], 'white',
'url(resources/images/symbol/grid.gif)');
var element = canvas.drawShape([200, 200], new OG.RectangleShape(), [150,
150]);
console.log(element);
```



출력된 element 객체를 살펴보면, 다음과 같은 Dom 객체로 이루어져 있는 것을 확인할 수 있습니다.

```
<g x="0" y="0" fill="none" stroke="#000" id="OG_4648_2" _type="SHAPE"
_shape="GEOM" style="-webkit-tap-highlight-color: rgba(0, 0, 0, 0); cursor:
move; position: relative;" _shape_id="OG.shape.RectangleShape">
    <path fill="#ffffff" stroke="#000000"
d="M125,125L275,125L275,275L125,275L125,125" style="-webkit-tap-highlight-
color: rgba(0, 0, 0, 0); fill-r=".5" fill-cx=".5" fill-cy=".5" fill-
opacity="0" id="OG_4648_3"></path><path fill="none" stroke="#fffff
d="M125,125L275,125L275,275L125,275L125,125" fill-opacity="0" stroke-width="20"
stroke-opacity="0" id="OG_4648_4" name="CONNECT_GUIDE" style="-webkit-tap-
highlight-color: rgba(0, 0, 0, 0); stroke-opacity: 0;">
    </path>
</g>
```

오픈그래프의 도형은 Dom 객체의 내용대로, g 태그 그룹이 있고 하위에 실제 화면에 표현되는 path / image / text svg 요소들로 이루어져 있습니다.

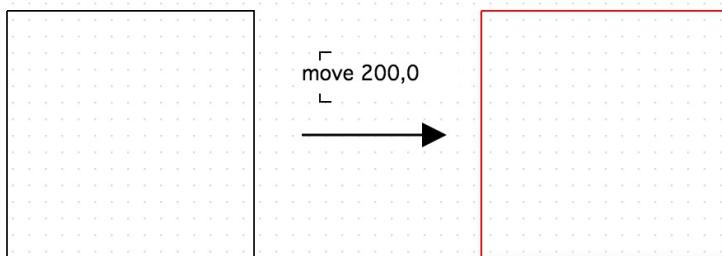
실제 화면에 표현되는 것은 g 태그 그룹 안의 내용이기 때문에, 화면 표현의 내용을 바꾸기 위해 element 의 요소를 직접 컨트롤 해도 적용되지 않습니다.

아래는 잘못된 접근 방식의 예제와 올바른 접근 방법의 예제입니다.

```
//잘못된 접근 방식: element 를 Jquery 로 직접 스타일 변경을 주어도 화면에 적용되지 않는다.  
var element = canvas.drawShape([200, 200], new OG.RectangleShape(), [150,  
150]);  
$(element).css('stroke', 'red');  
$(element).attr('x', 200);  
  
//올바른 접근 방식  
canvas.setShapeStyle(element, {'stroke': 'red'});  
canvas.move(element, [200, 0]);
```

위와 같은 효과를 내면서 다른방법으로 접근할 수 있는 방법을 알아보도록 합니다.

```
var element = canvas.drawShape([200, 200], new OG.RectangleShape(), [150,  
150]);  
  
element.shape.geom.style.map.stroke = 'red';  
element.shape.geom.move(200, 0);  
canvas.getRenderer().redrawShape(element);
```



```
canvas.setShapeStyle(element,  
{'stroke':'red'});  
canvas.move(element, [200,0]);
```

=

```
element.shape.geom.style.map.stroke = 'red';  
element.shape.geom.move(200,0);  
canvas.getRenderer().redrawShape(element);
```

위의 예제를 살펴보면, element 하위에 shape 객체가 있고, 다시 geom 객체, style 객체가 있는 것을 볼 수 있습니다.

각각의 객체를 출력해보고, 그 결과를 살펴봅시다.

```
var element = canvas.drawShape([200, 200], new OG.RectangleShape(), [150,  
150]);  
  
console.log(element.shape);  
console.log(element.shape.geom);  
console.log(element.shape.geom.style);
```

각각의 객체를 오픈그래프의 클래스에 대입하면 다음과 같습니다.

```
element.shape = OG.shape.RectangleShape  
element.shape.geom = OG.geometry.Rectangle  
element.shape.geom.style = OG.geometry.Style
```

오픈그래프의 고급 기능을 활용하고 협업의 다양한 요구에 수용하기 위해서는 위의 구조를 이해할 필요가 있습니다.

본 문서의 [Geometry \(./geometry.md\)](#), [Shape \(./shape.md\)](#), [Extend Shape \(./extend-shape.md\)](#) 페이지를 통해 더 자세히 살펴보도록 합니다.

## Event binding

위에서 오픈그래프의 도형의 Dom 객체를 통해 화면상의 스타일링에 변화를 줄 수 없는 것을 확인했습니다.

하지만 다른 UI 프레임워크를 활용한 이벤트 바인딩은 가능합니다.

이벤트 바인딩은 중첩될 수 있기 때문에, 오픈그래프 라이브러리가 이미 Dom에 바인딩한 경우라도 바인딩 요소를 추가할 수 있습니다.

### case

- 도형을 클릭시 선스타일과 채움색 변경

```
var element = canvas.drawShape([200, 200], new OG.RectangleShape(), [150,  
150]);  
$(element).click(function () {  
    canvas.setShapeStyle(this, {'stroke': 'red', fill: 'blue', 'fill-opacity':  
1})  
});
```

# Geometry

- [Point](#)
- [Coordinate](#)
- [Envelope](#)
- [Style](#)
- [Geometry](#)
- [GeometryCollection](#)
- [Polygon](#)
- [Rectangle](#)
- [PolyLine](#)
- [Line](#)
- [Curve](#)
- [BezierCurve](#)
- [Ellipse](#)
- [Circle](#)

Geometry 는 오픈그래프의 공간 기하 객체입니다.

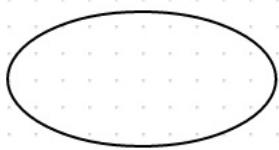
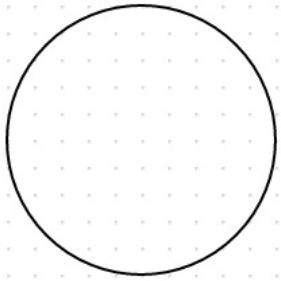
앞선 과정들로 오픈그래프 Dom 객체 하위에 도형 클래스가 있고, 도형 클래스 하위에 Geometry 공간 기하 객체가 있다 는 것을 기술했습니다.

도형 클래스는 Shape 이라고 표현하기도 합니다.

아래 예제에 몇가지 Shape 클래스를 호출하여 캔버스에 도형을 그려보도록 하겠습니다.

```
var circleShape = new OG.CircleShape();
var ellipseShape = new OG.EllipseShape();
var rectangleShape = new OG.RectangleShape();
var edgeShape1 = new OG.EdgeShape([50, 250], [200, 250]);
var edgeShape2 = new OG.EdgeShape([250, 250], [400, 250]);

var element1 = canvas.drawShape([100, 100], circleShape, [100, 100]);
var element2 = canvas.drawShape([250, 100], ellipseShape, [100, 50]);
var element3 = canvas.drawShape([400, 100], rectangleShape, [100, 50]);
var element4 = canvas.drawShape(null, edgeShape1, null, {
    'edge-type': 'plain',
    "arrow-start": "none",
    "arrow-end": "open-wide-long"
});
var element5 = canvas.drawShape(null, edgeShape2, null, {
    'edge-type': 'straight',
    "arrow-start": "classic-wide-long",
    "arrow-end": "block-wide-long",
    'stroke-dasharray': '-',
    'stroke-width': 4
});
```



각각의 Shape 가 Geometry 공간 기하 객체를 어떻게 호출하는지에 대해 살펴보기 위해 오픈그래프에서 제공하는 기본 Shape 중 하나인 CircleShape 의 코드를 살펴보도록 하겠습니다.

```

/**
 * Circle Shape
 *
 * @class
 * @extends OG.shape.GeoShape
 * @requires OG.common.*
 * @requires OG.geometry.*
 *
 * @param {String} label 라벨 [Optional]
 * @author <a href="mailto:sppark@uengine.org">Seungpil Park</a>
 */
OG.shape.CircleShape = function (label) {
    OG.shape.CircleShape.superclass.call(this);

    this.SHAPE_ID = 'OG.shape.CircleShape';
    this.label = label;
};

OG.shape.CircleShape.prototype = new OG.shape.GeoShape();
OG.shape.CircleShape.superclass = OG.shape.GeoShape;
OG.shape.CircleShape.prototype.constructor = OG.shape.CircleShape;
OG.CircleShape = OG.shape.CircleShape;

/**
 * 드로잉할 Shape 을 생성하여 반환한다.
 *
 * @return {OG.geometry.Geometry} Shape 정보
 * @override
 */
OG.shape.CircleShape.prototype.createShape = function () {
    if (this.geom) {
        return this.geom;
    }

    this.geom = new OG.geometry.Circle([50, 50], 50);
    return this.geom;
};

```

CircleShape 의 createShape 메소드의 코드를 다시 살펴보면, new OG.geometry.Circle 클래스를 반환해주고 있는 것을 볼 수 있습니다.

```

this.geom = new OG.geometry.Circle([50, 50], 50);
return this.geom;

```

Shape 의 createShape 메소드에 어떠한 geometry 공간 기하 객체를 삽입하는지에 따라 Shape 이 디스플레이 되는 모양이 달라지게 됩니다.

이를 사용하여 원 모양의 기하 객체와 그 원의 중심선을 지나는 선분을 가진 도형을 직접 제작해보도록 하겠습니다.

```

var canvas = new OG.Canvas('canvas', [1000, 800], 'white',
'url(resources/images/symbol/grid.gif')');

//CircleShape 를 선언.
var shape = new OG.CircleShape();

//CircleShape 의 createShape 을 오버라이딩 하여, 새로운 geometry 를 서술한다.
shape.createShape = function () {
    if (this.geom) {
        return this.geom;
    }

    var geom1, geom2, geomCollection = [];
    if (this.geom) {
        return this.geom;
    }

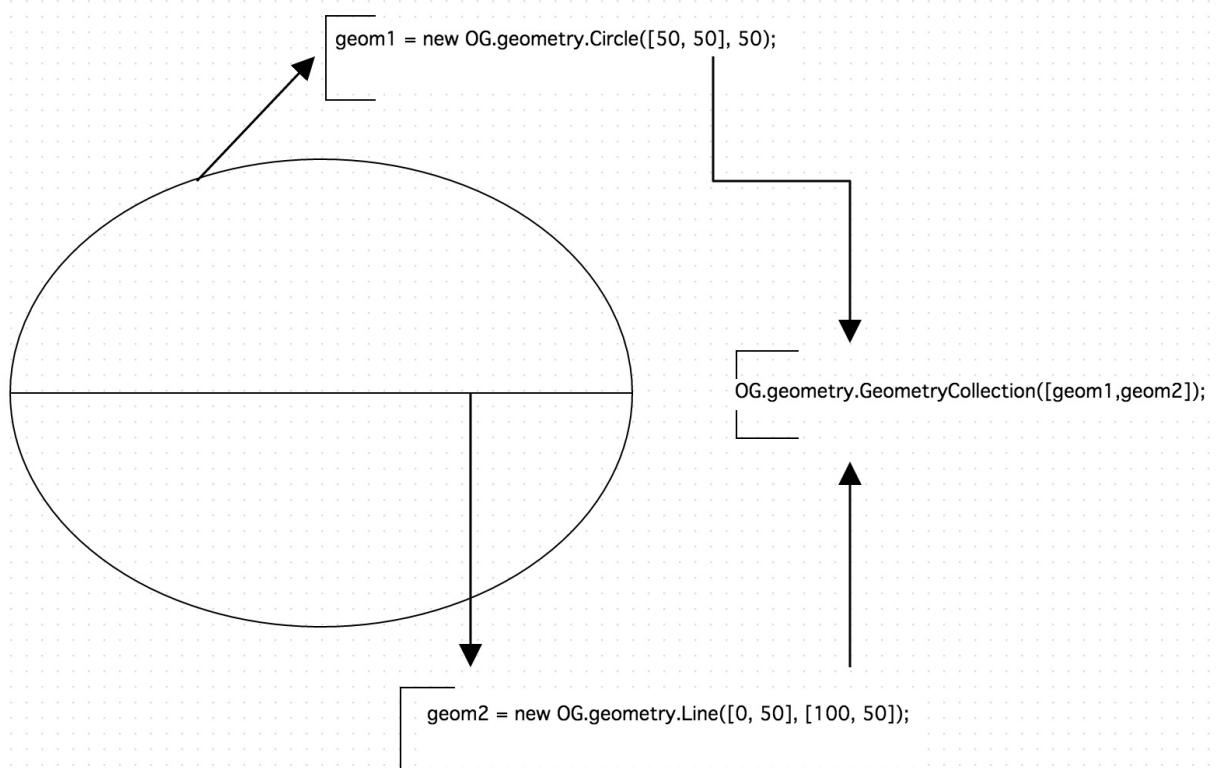
    geom1 = new OG.geometry.Circle([50, 50], 50);
    geom2 = new OG.geometry.Line([0, 50], [100, 50]);

    geomCollection.push(geom1);
    geomCollection.push(geom2);
    this.geom = new OG.geometry.GeometryCollection(geomCollection);

    return this.geom;
};

canvas.drawShape([400, 300], shape, [400, 300]);

```



예제에서는 GeometryCollection 이라는 geometry 가 사용되었습니다.

GeometryCollection 은 다수의 geometry 를 혼합하여 그려야 할 경우 사용됩니다.

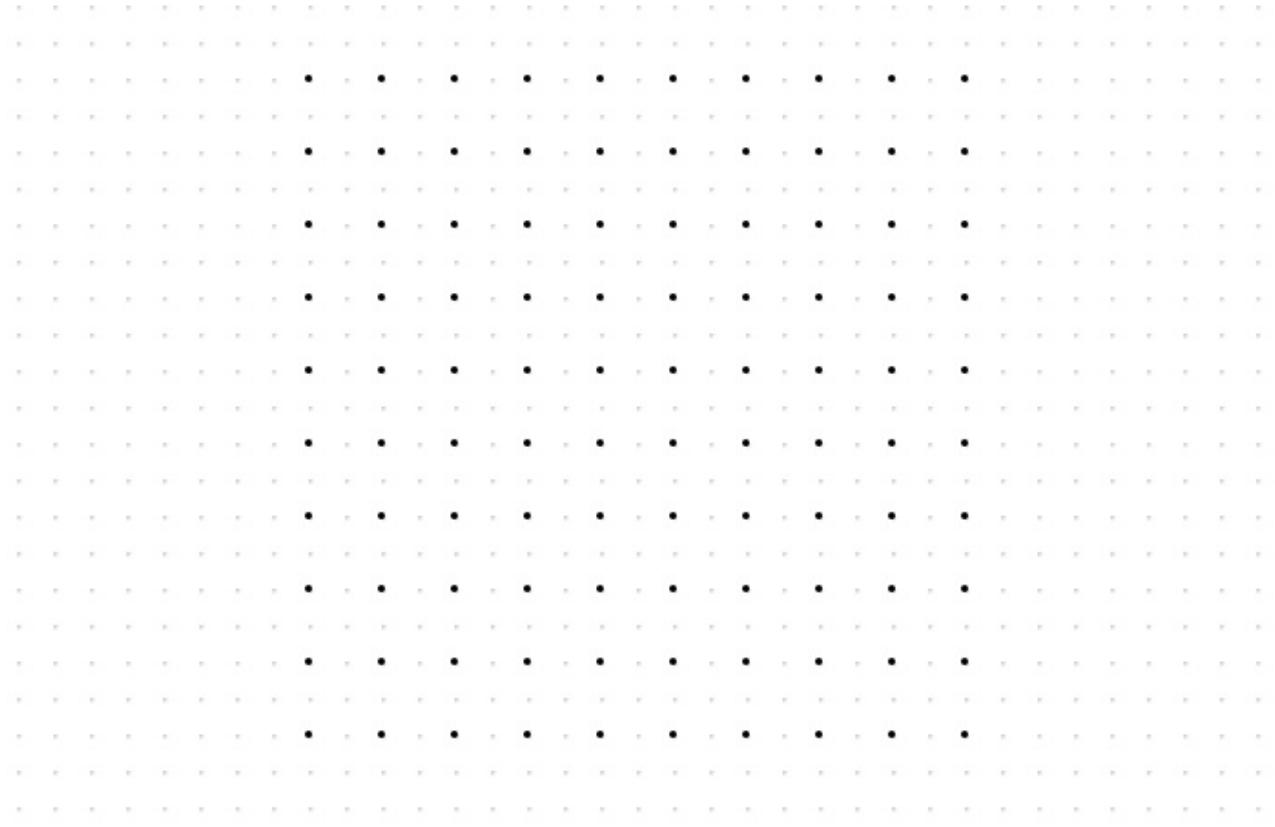
이어지는 내용은 오픈그래프에서 사용할 수 있는 geometry 목록과 그 사용법에 대한 기술입니다.

## Point

```
/**  
 * Point 공간 기하 객체(Spatial Geometry Object)  
 *  
 * @class  
 * @extends OG.geometry.Geometry  
 * @requires OG.geometry.Coordinate  
 * @requires OG.geometry.Envelope  
 * @requires OG.geometry.Geometry  
 *  
 * @example  
 * var geom = new OG.geometry.Point([20, 5]);  
 *  
 * @param {OG.geometry.Coordinate} coordinate 좌표값  
 * @author <a href="mailto:sppark@uengine.org">Seungpil Park</a>  
 */
```

### Example

```
for (var i = 0; i < 100; i++) {  
    var shape = new OG.GeoShape();  
    shape.SHAPETYPE = 'OG.shape.PointExample';  
    shape.createShape = function () {  
        if (this.geom) {  
            return this.geom;  
        }  
        this.geom = new OG.geometry.Point([0, 0]);  
        return this.geom;  
    };  
    var x = (i % 10) * 20 + 100;  
    var y = (Math.floor(i / 10) * 20) + 100;  
    console.log(x,y);  
    canvas.drawShape([x, y], shape, [0, 0]);  
}
```



## Coordinate

Coordinate 는 오픈그래프의 2차원 좌표계를 나타내는 기하 객체로써, 도형의 각 꼭지점을 표현하거나 Boundary 영역을 표현하는데 사용됩니다.

```
/**  
 * 2차원 좌표계에서의 좌표값  
 *  
 * @example  
 * var coordinate1 = new OG.geometry.Coordinate(10, 10);  
 * or  
 * var coordinate2 = new OG.geometry.Coordinate([20, 20]);  
 *  
 * @class  
 *  
 * @param {Number} x x좌표  
 * @param {Number} y y좌표  
 * @author <a href="mailto:sppark@uengine.org">Seungpil Park</a>  
 */
```

예제를 통해 다음의 두 도형을 그려보도록 합니다.

```

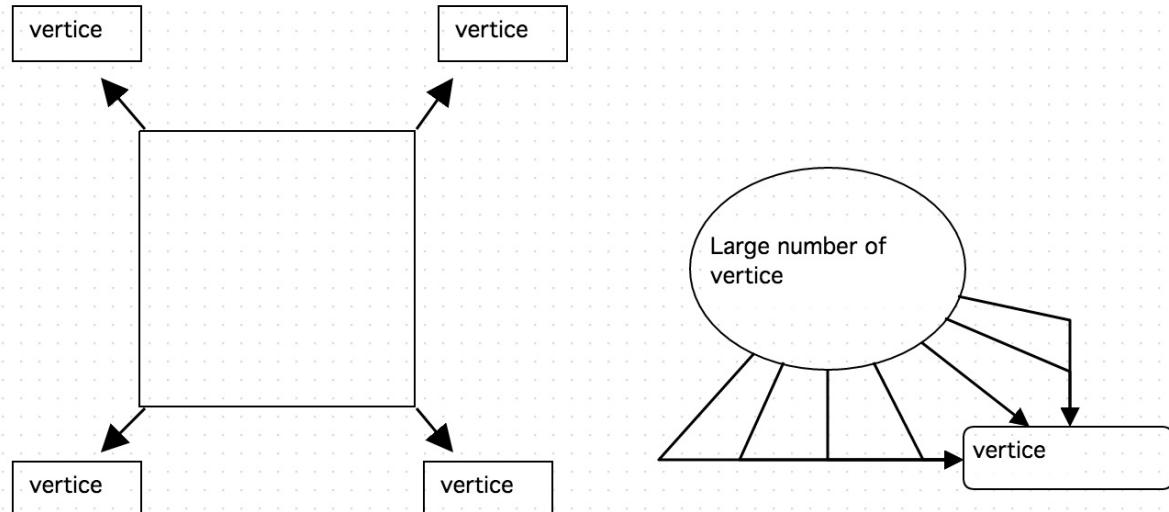
var rectElement = canvas.drawShape([200, 200], new OG.RectangleShape, [150, 150]);
var ellipseElement = canvas.drawShape([500, 200], new OG.EllipseShape, [150, 110]);

var rectVertices = rectElement.shape.geom.getVertices();
var ellipseVertices = ellipseElement.shape.geom.getVertices();

//print rectElement vertices
$.each(rectVertices, function(index, vertice){
    if(vertice instanceof OG.Coordinate){
        console.log(vertice.x , vertice.y);
    }
});

//print ellipseElement vertices
$.each(ellipseVertices, function(index, vertice){
    if(vertice instanceof OG.Coordinate){
        console.log(vertice.x , vertice.y);
    }
});

```



geometry 기하 객체 클래스의 getVertices 메소드는 기하객체가 가지고 있는 모든 꼭지점을 리턴합니다.

getVertices 메소드로 두 도형의 꼭지점들을 출력해보면,  
직사각형 모양의 rectElement 는 5개(선분이 사각형 모양을 그리기 위해서는 시작점으로 되돌아와야 하기 때문에 꼭지점이 하나 더 붙습니다.) 인 반면  
타원 모양의 ellipseElement 는 매우 많은 꼭지점들을 출력하는 것을 알 수 있습니다.

각 꼭지점의 리턴값은 OG.geometry.Coordinate 이며, x 프로퍼티와 y 프로퍼티로 값을 가져올 수 있습니다.

geometry 기하 객체의 vertices 를 이루고 있는 Coordinate 를 변경 또는 추가 할 경우 커스텀한 도형을 그릴 수 있습니다.

다음 예제를 통해 편행사변형과 랜덤한 변을 가진 사각형을 그려볼 수 있도록 합니다.

```

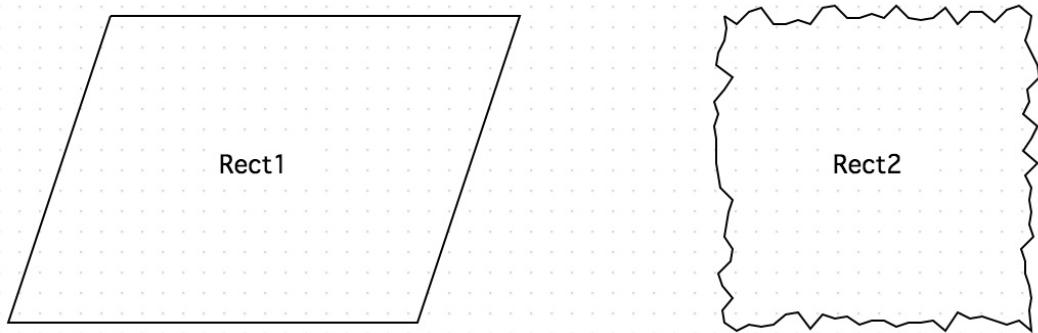
var element1 = canvas.drawShape([200, 200], new OG.RectangleShape('Rect1'),
[150, 150]);
var element2 = canvas.drawShape([500, 200], new OG.RectangleShape('Rect2'),
[150, 150]);

//평행사변형 만들기
var vertices = element1.shape.geom.getVertices();
vertices[1] = new OG.Coordinate(vertices[1].x + 50, vertices[1].y);
vertices[3] = new OG.Coordinate(vertices[3].x - 50, vertices[3].y);
canvas.getRenderer().redrawShape(element1);

//랜덤 모양 만들기
vertices = element2.shape.geom.getVertices();
var newVertices = [];

newVertices[0] = vertices[0];
newVertices[100] = vertices[4];
for (var i = 1; i < 100; i++) {
    var x, y;
    if (i < 25) {
        x = vertices[0].x + Math.floor(((vertices[1].x - vertices[0].x) / 25) *
i);
        y = vertices[0].y + Math.floor(Math.random() * 10) - 5;
        newVertices[i] = new OG.Coordinate(x, y);
    }
    else if (i < 50) {
        x = vertices[1].x + Math.floor(Math.random() * 10) - 5;
        y = vertices[1].y + Math.floor(((vertices[2].y - vertices[1].y) / 25) *
(i - 25));
        newVertices[i] = new OG.Coordinate(x, y);
    }
    else if (i < 75) {
        x = vertices[2].x - Math.floor(((vertices[2].x - vertices[3].x) / 25) *
(i - 50));
        y = vertices[2].y + Math.floor(Math.random() * 10) - 5;
        newVertices[i] = new OG.Coordinate(x, y);
    }
    else if (i < 100) {
        x = vertices[3].x + Math.floor(Math.random() * 10) - 5;
        y = vertices[3].y - Math.floor(((vertices[3].y - vertices[4].y) / 25) *
(i - 75));
        newVertices[i] = new OG.Coordinate(x, y);
    }
}
element2.shape.geom.vertices = newVertices;
canvas.getRenderer().redrawShape(element2);

```

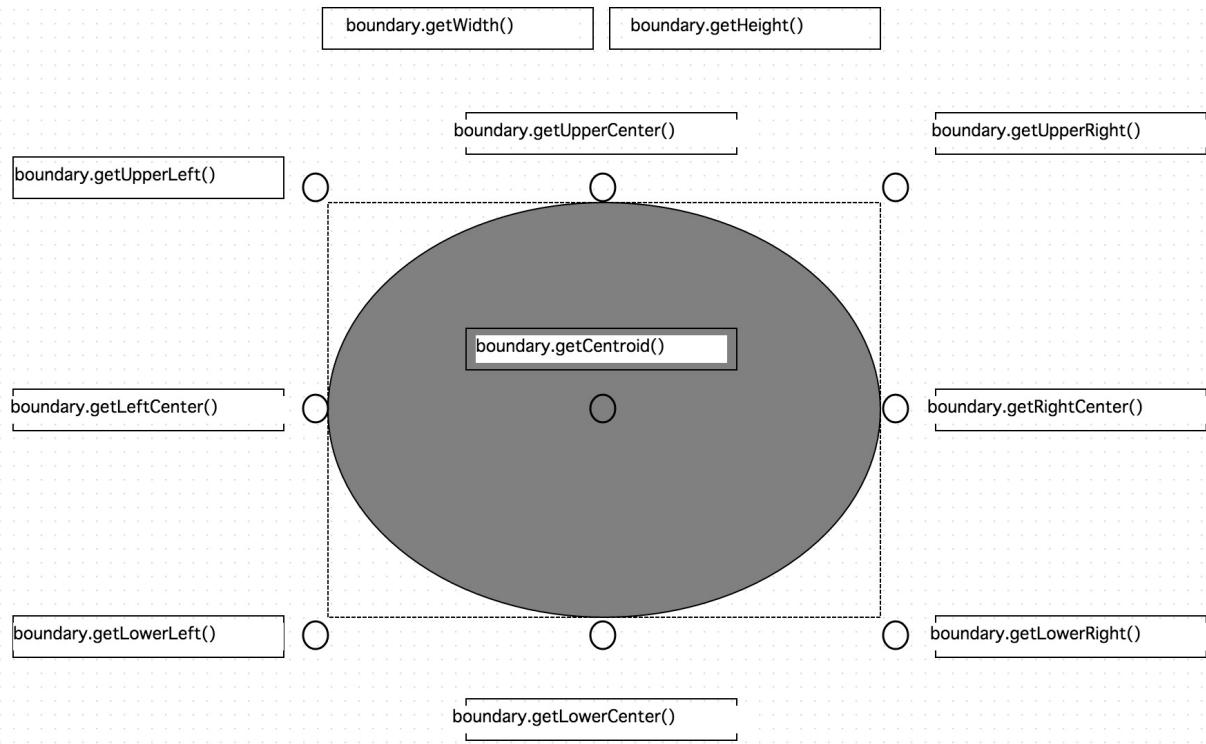


## Envelope

OG.geometry.Envelope 기하 객체는 앞선 [Canvas – getBoundary \(./canvas.md#getboundary\)](#) 파트에서 canvas.getBoundary() 메소드를 통해서도 접근할 수 있습니다.

canvas.getBoundary() 메소드를 호출하는 순간, 오픈그래프는 호출 대상이 어떠한 Shape 을 지니고 있던지 간에 도형의 모든 꼭지점을 포함시키는 최소한의 사각형의 영역을 만들어내어, OG.geometry.Envelope 기하 객체를 리턴합니다.

```
/**
 * 2차원 좌표계에서 Envelope 영역을 정의
 *
 * @class
 * @requires OG.geometry.Coordinate
 *
 * @example
 * var boundingBox = new OG.geometry.Envelope([50, 50], 200, 100);
 *
 * @param {OG.geometry.Coordinate|Number[]} upperLeft 기준 좌상단 좌표
 * @param {Number} width 너비
 * @param {Number} height 높이
 * @author <a href="mailto:sppark@uengine.org">Seungpil Park</a>
 */
```



## Style

OG.geometry.Style 은 모든 기하 객체의 style 프로퍼티에 위치하게 되며, 기하 객체의 svg 스타일을 정의합니다.

```
/**
 * 스타일(StyleSheet) Property 정보 클래스
 *
 * @class
 * @extends OG.common.HashMap
 *
 * @example
 * var style = new OG.geometry.Style({
 *   'cursor': 'default',
 *   'stroke': 'black'
 * });
 *
 * @param {Object} style 키:값 매핑된 스타일 프라퍼티 정보
 * @author <a href="mailto:sppark@uengine.org">Seungpil Park</a>
 */
```

OG.geometry.Style 을 응용한 Shape 렌더링 예제를 살펴보겠습니다.

```

var shape = new OG.CircleShape();
shape.createShape = function () {
    if (this.geom) {
        return this.geom;
    }

    var geom1, geom2, geomCollection = [];
    if (this.geom) {
        return this.geom;
    }

    geom1 = new OG.geometry.Circle([50, 50], 50);
    geom2 = new OG.geometry.Line([0, 50], [100, 50]);

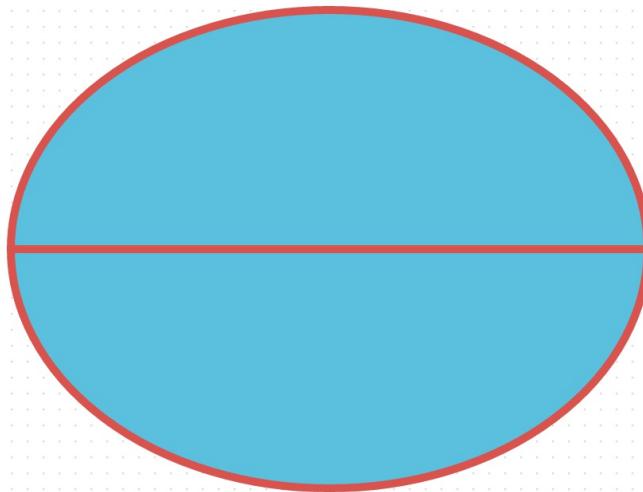
    geomCollection.push(geom1);
    geomCollection.push(geom2);
    this.geom = new OG.geometry.GeometryCollection(geomCollection);

    //geometry 의 스타일을 OG.geometry.Style 을 통해 정의한다.
    this.geom.style = new OG.geometry.Style({
        'cursor': 'default',
        'stroke': '#d9534f',
        'stroke-width': '5',
        'fill': '#5bc0de',
        'fill-opacity': 1
    });

    return this.geom;
};

canvas.drawShape([400, 300], shape, [400, 300]);

```



## Geometry

다음 한줄의 코드를 보겠습니다.

```
var geom = new OG.geometry.Circle([50, 50], 50);
```

지금 까지 나열되었던 Point, Coordinate, Envelope, Style 들은 위 코드의 OG.geometry.Circle 같은 geometry 기하 객체를 이루기 위한 부분적인 요소들이었습니다.

OG.geometry.Circle 같은 geometry 들은 경우에 따라 geometry 간의 상속을 반복할 수 있는데, 최상위 추상 클래스는 OG.geometry.Geometry 로써 항상 동일합니다.

OG.geometry.Geometry 최상위 추상 클래스는 2차원 좌표계에서의 공간 기하 계산을 위한 메소드 및 vertices(꼭지점 집합), style, boundary 영역 정보를 제공합니다.

아래 표는 OG.geometry.Geometry 가 제공하는 공간 기하 관련 메소드들입니다.

각 메소드의 인터페이스와 예제는 API Reference 문서를 참조하시길 바랍니다.

Method 명	기능
isEqual	주어진 Geometry 객체와 같은지 비교한다.
isContains	주어진 공간기하 객체를 포함하는지 비교한다.
isWithin	주어진 공간기하 객체에 포함되는지 비교한다.
getBoundary	공간기하 객체를 포함하는 사각형의 Boundary 영역을 반환한다.
getCentroid	공간기하 객체의 중심좌표를 반환한다.
getVertices	공간기하 객체의 모든 꼭지점을 반환한다.
minDistance	주어진 좌표와의 최단거리를 반환한다.
distance	주어진 공간기하 객체와의 중심점 간의 거리를 반환한다.
getLength	공간기하 객체의 길이를 반환한다.
move	가로, 세로 Offset 만큼 좌표를 이동한다.
moveCentroid	주어진 중심좌표로 공간기하 객체를 이동한다.
resize	상, 하, 좌, 우 외곽선을 이동하여 Envelope 을 리사이즈 한다.
resizeBox	중심좌표는 고정한 채 Bounding Box 의 width, height 를 리사이즈 한다.
rotate	기준 좌표를 기준으로 주어진 각도 만큼 회전한다.
fitToBoundary	주어진 Boundary 영역 안으로 공간 기하 객체를 적용한다.(이동 & 리사이즈)
convertCoordinate	파라미터가 [x, y] 형식의 좌표 Array 이면 OG.geometry.Coordinate 인스턴스를 new 하여 반환한다.
distanceToLine	포인트 P로부터 라인 AB의 거리를 계산한다.
distanceLineToLine	라인1로부터 라인2의 거리를 계산한다.
intersectToLine	주어진 라인과 교차하는 좌표를 반환한다.
shortestIntersectToLine	주어진 라인과 교차하는 좌표 중 시작좌표에 가장 가까운 좌표를 반환한다.
intersectLineToLine	라인1로부터 라인2의 교차점을 계산한다.
intersectCircleToLine	라인1로부터 라인2의 교차점을 계산한다.
intersectPointToLine	포인트 P로부터 라인 AB의 교차점을 계산한다.
getPercentageDistanceFromPoint	주어진 좌표에 대해 공간기하 객체의 퍼센테이지 비율을 구한다.
isContainsPoint	공간기하 객체가 주어진 좌표를 포함하는지를 반환한다.
getPointFromPercentageDistance	공간기하 객체에 대한 퍼센테이지 좌표의 실제 좌표를 구한다.
reset	저장된 boundary 를 클리어하여 새로 계산하도록 한다.

## GeometryCollection

GeometryCollection 은 다수의 Geometry 를 묶어 하나의 Geometry 로 제공하는 클래스입니다.

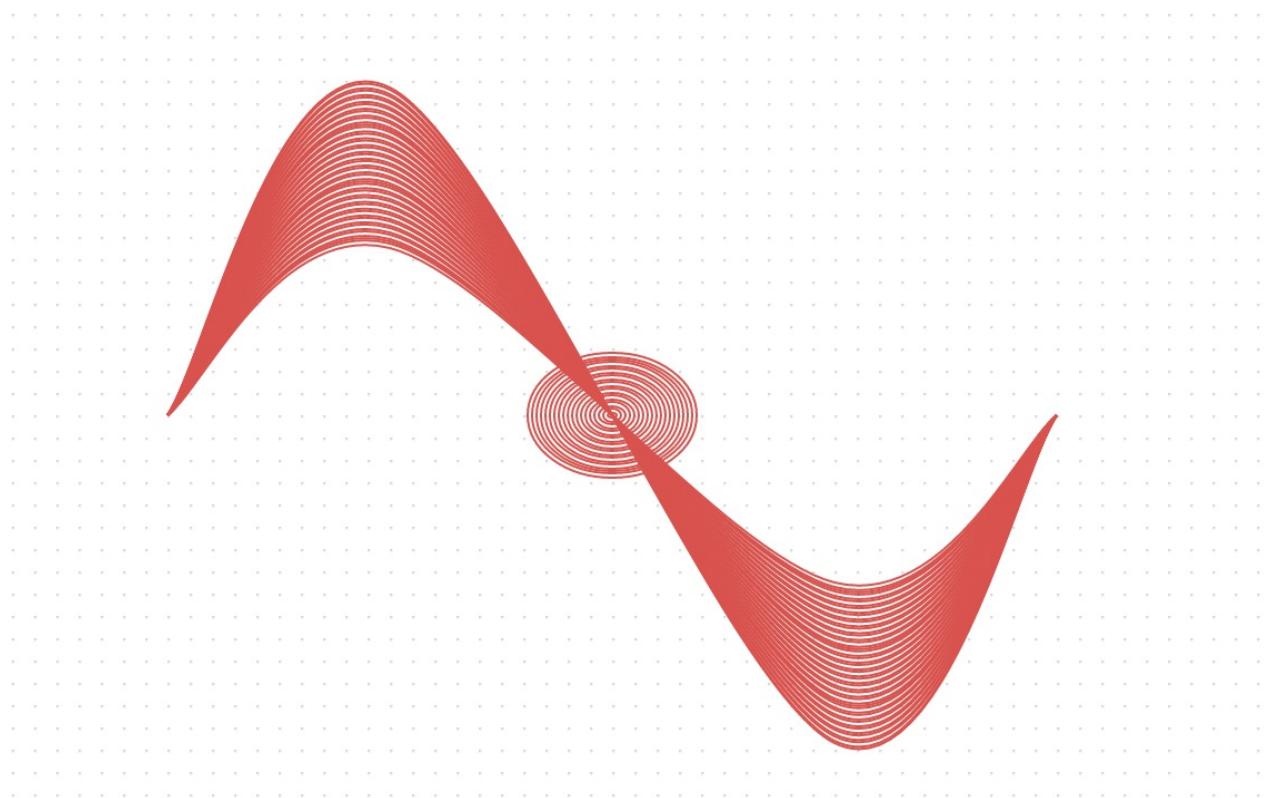
new OG.geometry.GeometryCollection([geom1, geom2, geom3]) 식으로 호출하여 씁니다.

```
/**  
 * 공간 기하 객체(Spatial Geometry Object) Collection  
 *  
 * @class  
 * @extends OG.geometry.Geometry  
 * @requires OG.geometry.Coordinate  
 * @requires OG.geometry.Envelope  
 * @requires OG.geometry.Geometry  
 *  
 * @example  
 * var geom1 = new OG.geometry.Point([20, 5]),  
 *         geom2 = new OG.geometry.Line([20, 5], [30, 15]),  
 *         geom3 = new OG.geometry.PolyLine([[20, 5], [30, 15], [40, 25], [50,  
15]]);  
 *  
 * var collection = new OG.geometry.GeometryCollection([geom1, geom2, geom3]);  
 *  
 * @param geometries {OG.geometry.Geometry[]} 공간 기하 객체 Array  
 * @author <a href="mailto:sppark@uengine.org">Seungpil Park</a>  
 */
```

Example)

```
var shape = new OG.GeoShape();  
shape.SHAP_ID = 'OG.shape.CustomCollection';  
shape.createShape = function () {  
    if (this.geom) {  
        return this.geom;  
    }  
  
    var geomCollection = [];  
    if (this.geom) {  
        return this.geom;  
    }  
  
    for (var i = 0; i < 50; i++) {  
        var geom = new OG.geometry.Curve([[200, 100], [100, 300 - i * 2],  
[-100, -100 + i * 2], [-200, 100]]);  
        geomCollection.push(geom);  
    }  
    for (var i = 0; i < 20; i++) {  
        geomCollection.push(new OG.geometry.Circle([0, 100], i * 2));  
    }  
  
    this.geom = new OG.geometry.GeometryCollection(geomCollection);  
  
    this.geom.style = new OG.geometry.Style({  
        'cursor': 'default',  
        'stroke': '#d9534f',
```

```
'stroke-width': '1',
'fill': 'none',
'fill-opacity': 0
});
return this.geom;
};
var customShape = canvas.drawShape([400, 300], shape, [400, 300]);
```



## Polygon

Polygon 은 닫힌 선분 도형을 그리는 클래스이며, new OG.geometry.Polygon([[x,y],[x,y][x,y]...]) 으로 호출할 수 있습니다.

닫힌 선분 도형이기 때문에 시작점과 끝점은 같아야 합니다.

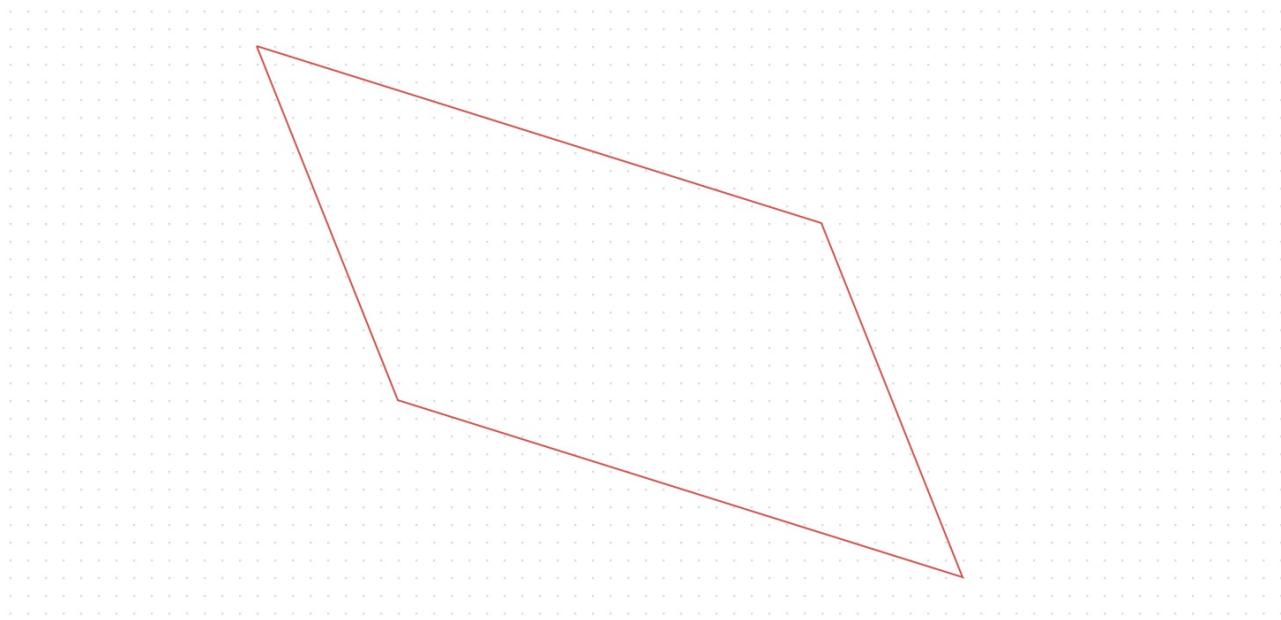
```
/**
 * Polygon 공간 기하 객체(Spatial Geometry Object)
 *
 * @class
 * @extends OG.geometry.PolyLine
 * @requires OG.geometry.Coordinate
 * @requires OG.geometry.Envelope
 * @requires OG.geometry.Geometry
 *
 * @example
 * var geom = new OG.geometry.Polygon([[20, 5], [30, 15], [40, 25], [50, 15],
 * [60, 5], [20, 5]]);
 *
 * @param {OG.geometry.Coordinate[]} vertices Line Vertex 좌표 Array
 * @author <a href="mailto:sppark@uengine.org">Seungpil Park</a>
 */

```

Example)

```
var shape = new OG.GeoShape();
shape.SHAP_ID = 'OG.shape.PolygonExample';
shape.createShape = function () {
    if (this.geom) {
        return this.geom;
    }
    this.geom = new OG.geometry.Polygon([[0, 0], [200, 50], [250, 150], [50,
100], [0, 0]]);

    this.geom.style = new OG.geometry.Style({
        'cursor': 'default',
        'stroke': '#d9534f',
        'stroke-width': '1',
        'fill': 'none',
        'fill-opacity': 0
    });
    return this.geom;
};
var customShape = canvas.drawShape([400, 300], shape, [400, 300]);
```



## Rectangle

OG.geometry.Rectangle 클래스는 OG.geometry.Polygon 을 상속받아 좌측상단의 시작점, 가로, 세로 세가지의 값으로 직사각형을 그려줍니다.

오픈그래프에서 지원하는 Shape 종류에는 mageShape, TextShape, HtmlShape 등과 같이 기하 도형으로 이루어지지 않은 도형들이 있는데,

이 Shape 들은 기본적으로 OG.geometry.Rectangle 를 Shape 의 geometry 로서 가져갑니다.

```

/**
 * Rectangle 공간 기하 객체(Spatial Geometry Object)
 *
 * @class
 * @extends OG.geometry.Polygon
 * @requires OG.geometry.Coordinate
 * @requires OG.geometry.Envelope
 * @requires OG.geometry.Geometry
 *
 * @example
 * var geom = new OG.geometry.Rectangle([20, 5], 10, 10);
 *
 * @param {OG.geometry.Coordinate} upperLeft 좌상단좌표
 * @param {Number} width 너비
 * @param {Number} height 높이
 * @author <a href="mailto:sppark@uengine.org">Seungpil Park</a>
 */

```

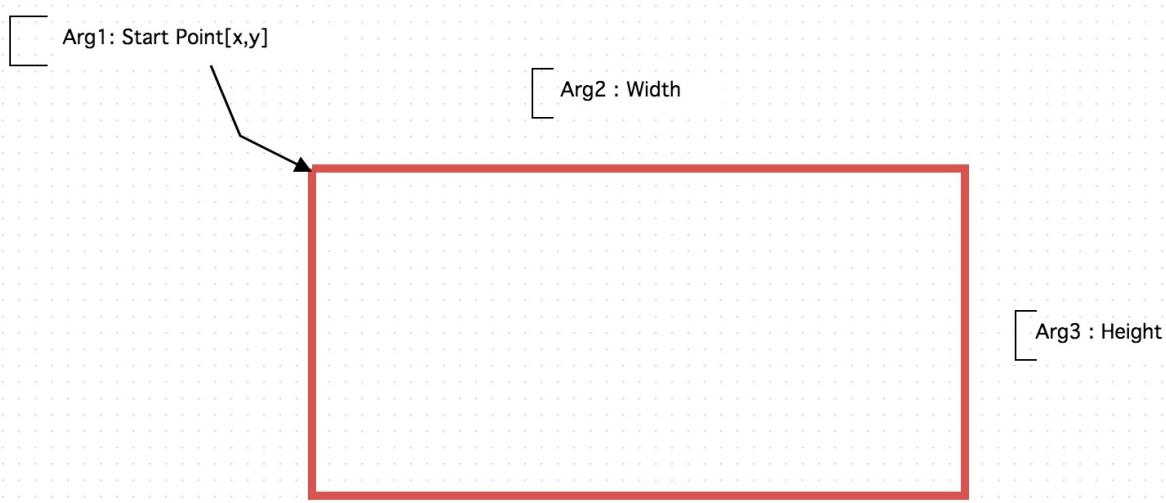
Example)

```

var shape = new OG.GeoShape();
shape.SHAPES_ID = 'OG.shape.RectExample';
shape.createShape = function () {
    if (this.geom) {
        return this.geom;
    }
    this.geom = new OG.geometry.Rectangle([0,0],200,50);

    this.geom.style = new OG.geometry.Style({
        'cursor': 'default',
        'stroke': '#d9534f',
        'stroke-width': '5',
        'fill': 'none',
        'fill-opacity': 0
    });
    return this.geom;
};
var customShape = canvas.drawShape([400, 300], shape, [400, 200]);

```



## PolyLine

OG.geometry.PolyLine 은 꺽은선 선분 기하 클래스이며 OG.geometry.PolyLine 을 extends 한 클래스는 다음 것들이 있습니다.

- OG.geometry.BezierCurve
- OG.geometry.Curve
- OG.geometry.Line
- OG.geometry.Polygon

아래 표는 OG.geometry.PolyLine 가 제공하는 공간 기하 관련 메소드들입니다.

각 메소드의 인터페이스와 예제는 API Reference 문서를 참조하시길 바랍니다.

Method 명	기능
getVertices	공간기하 객체의 모든 꼭지점을 반환한다.
setVertices	공간기하 객체의 꼭지점을 설정한다.
move	가로, 세로 Offset 만큼 좌표를 이동한다.
resize	상, 하, 좌, 우 외곽선을 이동하여 Envelope 을 리사이즈 한다.
rotate	기준 좌표를 기준으로 주어진 각도 만큼 회전한다.
toString	객체 프라퍼티 정보를 JSON 스트링으로 반환한다.
angleBetweenPoints	공간기하 객체의 두 꼭지점 사이에 가상의 선을 그렸을 때, 그 기울기를 구한다.
isRightAngleBetweenPoints	공간기하 객체의 두 꼭지점 사이의 기울기가 수평 또는 수직인지 판별한다.
angleBetweenThreePoints	공간기하 객체의 세 꼭지점 사이의 각도 중 작은 각도를 반환한다.

```
/**
 * PolyLine 공간 기하 객체(Spatial Geometry Object)
 *
 * @class
 * @extends OG.geometry.Geometry
 * @requires OG.geometry.Coordinate
 * @requires OG.geometry.Envelope
 * @requires OG.geometry.Geometry
 *
 * @example

```

```

* var geom = new OG.geometry.PolyLine([[20, 5], [30, 15], [40, 25], [50,
15]]);
*
* @param {OG.geometry.Coordinate[]} vertices Line Vertex 좌표 Array
* @author <a href="mailto:sppark@uengine.org">Seungpil Park</a>
*/

```

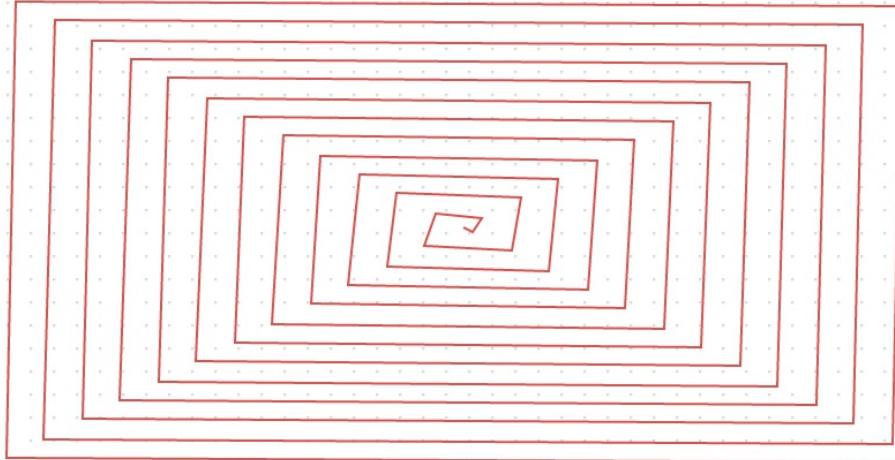
Example)

```

var shape = new OG.GeoShape();
shape.SHAP_ID = 'OG.shape.PolyLineExample';
shape.createShape = function () {
    if (this.geom) {
        return this.geom;
    }
    var vertices = [];
    var distancs = 40;
    for(var i = 0 ; i < 50; i++){
        if(i % 4 == 0){
            vertices[i] = [ - Math.floor((i / 4) * distancs) , Math.floor((i /
4) * distancs)]
        }else if(i % 4 == 1){
            vertices[i] = [ Math.floor((i / 4) * distancs) , Math.floor((i / 4)
* distancs)]
        }else if(i % 4 == 2){
            vertices[i] = [ Math.floor((i / 4) * distancs) , - Math.floor((i /
4) * distancs)]
        }else{
            vertices[i] = [ -Math.floor((i / 4) * distancs) , - Math.floor((i /
4) * distancs)]
        }
    }
    this.geom = new OG.geometry.PolyLine(vertices);

    this.geom.style = new OG.geometry.Style({
        'cursor': 'default',
        'stroke': '#d9534f',
        'stroke-width': '1',
        'fill': 'none',
        'fill-opacity': 0
    });
    return this.geom;
};
var customShape = canvas.drawShape([400, 300], shape, [400, 200]);

```



## Line

OG.geometry.Line OG.geometry.PolyLine 을 extends 한 클래스로서 시작과 끝점을 받아 직선을 표현합니다.

```
/**  
 * Line 공간 기하 객체(Spatial Geometry Object)  
 *  
 * @class  
 * @extends OG.geometry.PolyLine  
 * @requires OG.geometry.Coordinate  
 * @requires OG.geometry.Envelope  
 * @requires OG.geometry.Geometry  
 *  
 * @example  
 * var geom = new OG.geometry.Line([20, 5], [30, 15]);  
 *  
 * @param {OG.geometry.Coordinate} from 라인 시작 좌표값  
 * @param {OG.geometry.Coordinate} to 라인 끝 좌표값  
 * @author <a href="mailto:sppark@uengine.org">Seungpil Park</a>  
 */
```

Example)

```
var geom = new OG.geometry.Line([20, 5], [30, 15]);
```

## Curve

OG.geometry.Curve 는 OG.geometry.PolyLine 을 extends 한 클래스로서 모든 콘트롤 포인트를 공유하는 곡선을 표현합니다.

```
/**  
 * Catmull-Rom Spline Curve 공간 기하 객체(Spatial Geometry Object)  
 * 모든 콘트롤포인트를 지나는 곡선을 나타낸다.  
 */
```

```

* @class
* @extends OG.geometry.PolyLine
* @requires OG.geometry.Coordinate
* @requires OG.geometry.Envelope
* @requires OG.geometry.Geometry
* @requires OG.common.CurveUtil
*
* @example
* var geom = new OG.geometry.Curve([[200, 100], [100, 300], [-100, -100],
[-200, 100]]);
*
* @param {OG.geometry.Coordinate[]} controlPoints Curve Vertex 좌표 Array
* @author <a href="mailto:sppark@uengine.org">Seungpil Park</a>
*/

```

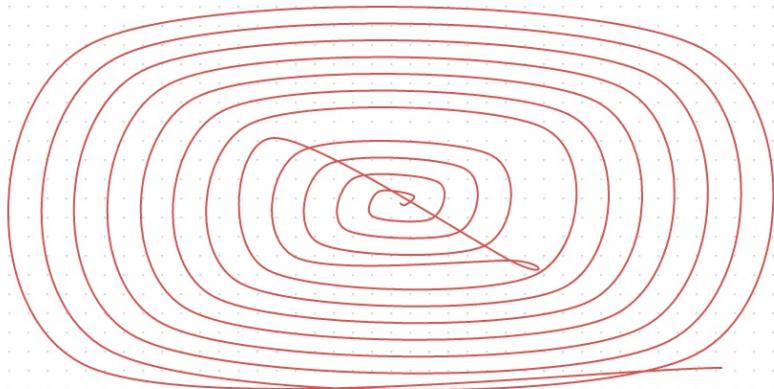
Example)

```

var shape = new OG.GeoShape();
shape.SHAPES_ID = 'OG.shape.PolyLineExample';
shape.createShape = function () {
    if (this.geom) {
        return this.geom;
    }
    var vertices = [];
    var distancs = 40;
    for(var i = 0 ; i < 50; i++){
        if(i % 4 == 0){
            vertices[i] = [ - Math.floor((i / 4) * distancs) , Math.floor((i /
4) * distancs)]
        }else if(i % 4 == 1){
            vertices[i] = [ Math.floor((i / 4) * distancs) , Math.floor((i / 4)
* distancs)]
        }else if(i % 4 == 2){
            vertices[i] = [ Math.floor((i / 4) * distancs) , - Math.floor((i /
4) * distancs)]
        }else{
            vertices[i] = [ -Math.floor((i / 4) * distancs) , - Math.floor((i /
4) * distancs)]
        }
    }
    this.geom = new OG.geometry.Curve(vertices);

    this.geom.style = new OG.geometry.Style({
        'cursor': 'default',
        'stroke': '#d9534f',
        'stroke-width': '1',
        'fill': 'none',
        'fill-opacity': 0
    });
    return this.geom;
};
var customShape = canvas.drawShape([400, 300], shape, [400, 200]);

```



## BezierCurve

OG.geometry.Curve 는 OG.geometry.PolyLine 을 extends 한 클래스로서 콘트롤팝인트1, 콘트롤팝인트2에 의해 시작좌표, 끝좌표를 지나는 곡선을 나타냅니다.

new OG.geometry.BezierCurve([시작좌표, 콘트롤팝인트1, 콘트롤팝인트2, 끝좌표]) 로 호출하며, 인자가 4개가 아닐 경우 Exception 이 발생합니다.

```

/**
 * Cubic Bezier Curve 공간 기하 객체(Spatial Geometry Object)
 * 콘트롤포인트1, 콘트롤포인트2에 의해 시작좌표, 끝좌표를 지나는 곡선을 나타낸다.
 *
 * @class
 * @extends OG.geometry.PolyLine
 * @requires OG.geometry.Coordinate
 * @requires OG.geometry.Envelope
 * @requires OG.geometry.Geometry
 * @requires OG.common.CurveUtil
 *
 * @example
 * var geom = new OG.geometry.BezierCurve([[200, 100], [100, 300], [-100,
 -100], [-200, 100]]);
 *
 * @param {OG.geometry.Coordinate[]} controlPoints [from, control_point1,
 control_point2, to]
 * @author <a href="mailto:sppark@uengine.org">Seungpil Park</a>
 */

```

Example)

```

var shape = new OG.EdgeShape();
shape.SHAPE_ID = 'OG.shape.BezierCurveExample';
shape.createShape = function () {
    if (this.geom) {
        return this.geom;
    }
    this.geom = new OG.geometry.BezierCurve([-200, 100], [-100, 0], [100,
200], [200, 100]);

    this.geom.style = new OG.geometry.Style({
        'cursor': 'default',
        'stroke': '#d9534f',
        'stroke-width': '3',
        'fill': 'none',
        'fill-opacity': 0
    });
    return this.geom;
};
var customShape = canvas.drawShape([300, 100], shape);

```



## Ellipse

OG.geometry.Ellipse 는 OG.geometry.Curve 를 extends 한 클래스로서 타원형을 그립니다.

new OG.geometry.Ellipse(중심좌표, x 축 반경, y 축 반경, x 축 기울기) 로 호출합니다.

```

/**
 * Ellipse 공간 기하 객체(Spatial Geometry Object)
 *
 * @class
 * @extends OG.geometry.Curve
 * @requires OG.geometry.Coordinate
 * @requires OG.geometry.Envelope
 * @requires OG.geometry.Geometry
 *
 * @example
 * var geom = new OG.geometry.Ellipse([10, 10], 10, 5);
 *
 * @param {OG.geometry.Coordinate} center Ellipse 중심 좌표
 * @param {Number} radiusX X축 반경
 * @param {Number} radiusY Y축 반경
 * @param {Number} angle X축 기울기
 * @author <a href="mailto:sppark@uengine.org">Seungpil Park</a>
 */

```

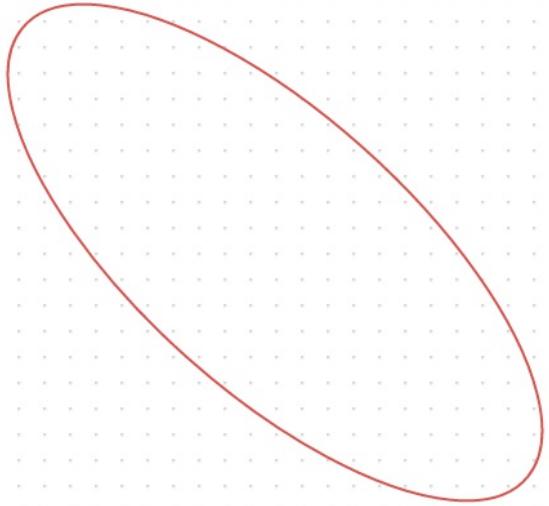
Example)

```

var shape = new OG.GeoShape();
shape.SHAP_ID = 'OG.shape.EllipseExample';
shape.createShape = function () {
    if (this.geom) {
        return this.geom;
    }
    this.geom = new OG.geometry.Ellipse([0, 0], 300, 100, 30);

    this.geom.style = new OG.geometry.Style({
        'cursor': 'default',
        'stroke': '#d9534f',
        'stroke-width': '1',
        'fill': 'none',
        'fill-opacity': 0
    });
    return this.geom;
};
var customShape = canvas.drawShape([300, 200], shape, [200, 200]);

```



## Circle

OG.geometry.Circle 은 OG.geometry.Ellipse 를 extends 한 클래스로서 원형을 그립니다.

new OG.geometry.Circle(중심좌표, 반경) 으로 호출합니다.

```
/**
 * Circle 공간 기하 객체(Spatial Geometry Object)
 *
 * @class
 * @extends OG.geometry.Ellipse
 * @requires OG.geometry.Coordinate
 * @requires OG.geometry.Envelope
 * @requires OG.geometry.Geometry
 *
 * @example
 * var geom = new OG.geometry.Circle([10, 10], 5);
 *
 * @param {OG.geometry.Coordinate} center Circle 중심 좌표
 * @param {Number} radius radius 반경
 * @author <a href="mailto:sppark@uengine.org">Seungpil Park</a>
 */

```

Example)

```
var shape = new OG.GeoShape();
shape.SHAP_ID = 'OG.shape.CircleExample';
shape.createShape = function () {
    if (this.geom) {
        return this.geom;
    }
    this.geom = new OG.geometry.Circle([50, 50], 100);

    this.geom.style = new OG.geometry.Style({
        'cursor': 'default',
        'stroke': '#d9534f',
        'stroke-width': '1',
        'fill': 'none',
        'fill-opacity': 0
    });
    return this.geom;
};
var customShape = canvas.drawShape([300, 200], shape, [200, 200]);
```

# Shape

- [IShape](#)
- [Configuration](#)
- [GeomShape](#)
- [EdgeShape](#)
- [GroupShape](#)
- [ImageShape](#)
- [TextShape](#)

Shape 은 오픈그래프 캔버스에 drawing 하기 위한 도형을 정의한 클래스입니다.

Shape 은 화면에 표현될 Geometry 클래스를 가지고 있고, 사용자가 UI 상에서 도형을 선택하거나 이동시키는 등의 이벤트에 대한 제어 내용을 포함하고 있습니다.

Shape 에 대한 접근은 도형 Dom 객체의 shape 프로퍼티를 통해 가져올 수 있습니다.

```
var canvas = new OG.Canvas('canvas', [1000, 800], 'white',
  'url(resources/images/symbol/grid.gif')';

//Shape 정의
var shape = new OG.RectangleShape();

//Shape 을 화면에 drawing 하여 Dom 객체를 얻음.
var element = canvas.drawShape([400, 300], new OG.RectangleShape(), [400,
  300]);

//Dom 객체의 shape 프로퍼티에 등록되어있음.
element.shape ==> Shape 클래스
```

## IShape

OG.shape.IShape 는 Shape 의 최상위 추상 클래스입니다.

OG.shape.IShape 에는 도형, 텍스트, 이미지 등의 드로잉 될 Object 의 정보를 저장과 이벤트 제어를 담당합니다.

OG.shape.IShape 의 소스코드를 본 후, Shape 클래스의 Configuration 설정 방법을 알아보도록 하겠습니다.

```
/**
 * 도형, 텍스트, 이미지 등의 드로잉 될 Object 의 정보를 저장하는 Shape 정보 최상위 인터페이스
 *
 * @class
 * @requires OG.common.*
 * @requires OG.geometry.*
 *
 * @author <a href="mailto:sppark@uengine.org">Seungpil Park</a>
 */
OG.shape.IShape = function () {
  /**
   * Shape 유형(GEOM, TEXT, HTML, IMAGE, EDGE, GROUP)
   * @type String
```

```
 */
this.TYPE = null;

/**
 * Shape 을 구분하는 Shape ID(Shape 클래스명과 일치)
 * @type String
 */
this.SHAPE_ID = null;

/**
 * Shape 모양을 나타내는 공간기하 객체(Geometry)
 * @type OG.geometry.Geometry
 */
this.geom = null;

/**
 * Shape 라벨 텍스트
 * @type String
 */
this.label = null;

/**
 * Shape 의 Collapse 여부
 * @type Boolean
 */
this.isCollapsed = false;

// 이벤트 속성
/**
 * 선택 가능여부
 * @type Boolean
 */
this.SELECTABLE = true;

/**
 * 이동 가능여부
 * @type Boolean
 */
this.MOVABLE = true;

/**
 * 리사이즈 가능여부
 * @type Boolean
 */
this.RESIZABLE = true;

/**
 * 연결 가능여부
 * @type Boolean
 */
this.CONNECTABLE = true;
```

```
/**  
 * From 연결 가능여부 (From(Shape) => To)  
 * @type Boolean  
 */  
this.ENABLE_FROM = true;  
  
/**  
 * To 연결 가능여부 (From => To(Shape))  
 * @type Boolean  
 */  
this.ENABLE_TO = true;  
  
/**  
 * T0 연결 가능여부  
 * @type Boolean  
 */  
this.ENABLE_FROM = true;  
  
/**  
 * Self 연결 가능여부  
 * @type Boolean  
 */  
this.SELF_CONNECTABLE = true;  
  
/**  
 * 가이드에 자기자신을 복사하는 컨트롤러 여부.  
 * @type Boolean  
 */  
this.CONNECT_CLONEABLE = true;  
  
/**  
 * 드래그하여 연결시 연결대상 있는 경우에만 Edge 드로잉 처리 여부  
 * @type Boolean  
 */  
this.CONNECT_REQUIRED = true;  
  
/**  
 * 드래그하여 연결시 그룹을 건너뛸때 스타일 변경 여부  
 * @type Boolean  
 */  
this.CONNECT_STYLE_CHANGE = true;  
  
/**  
 * 가이드에 삭제 컨트롤러 여부  
 * @type Boolean  
 */  
this.DELETEABLE = true;  
  
/**  
 * 라벨 수정여부  
 * @type Boolean  
 */
```

```

        */
        this.LABEL_EDITABLE = true;

    }

    this.exceptionType = '';
};

OG.shape.IShape.prototype = {

    /**
     * 드로잉할 Shape 를 생성하여 반환한다.
     *
     * @return {*} Shape 정보
     * @abstract
     */
    createShape: function () {
        throw new OG.NotImplementedException("OG.shape.IShape.createShape");
    },

    /**
     * Shape 을 복사하여 새로운 인스턴스로 반환한다.
     *
     * @return {OG.shape.IShape} 복사된 인스턴스
     * @abstract
     */
    clone: function () {
        throw new OG.NotImplementedException("OG.shape.IShape.clone");
    },
    addEve : function(){}
    ,

    // (void) 특수한 컨트롤을 생성하기 위한 함수
    drawCustomControl: function(){}
}

};

OG.shape.IShape.prototype.constructor = OG.shape.IShape;
OG.IShape = OG.shape.IShape;

```

## Configuration

아래 표는 OG.shape.IShape 가 제공하는 이벤트 핸들링 Configuration 및 Object 정보를 담당하는 프로퍼티 리스트입니다.

Property 명	역할	타입
TYPE	Shape 유형(GEOM, TEXT, HTML, IMAGE, EDGE, GROUP)	String
SHAPE_ID	Shape 을 구분하는 Shape ID(Shape 클래스명과 일치)	String
geom	Shape 모양을 나타내는 공간기하 객체(Geometry)	OG.geometry.Geometry
label	Shape 라벨 텍스트	String
isCollapsed	Shape 의 Collapse 여부	Boolean
SELECTABLE	선택 가능여부	Boolean

MOVABLE	이동 가능여부	Boolean
RESIZABLE	리사이즈 가능여부	Boolean
CONNECTABLE	연결 가능여부	Boolean
ENABLE_FROM	From 연결 가능여부 (From(Shape) => To)	Boolean
ENABLE_TO	To 연결 가능여부 (From => To(Shape))	Boolean
SELF_CONNECTABLE	Self 연결 가능여부	Boolean
CONNECT_CLONEABLE	가이드에 자기자신을 복사하는 컨트롤러 여부.	Boolean
CONNECT_REQUIRED	드래그하여 연결시 연결대상 있는 경우에만 Edge 드로잉 처리 여부	Boolean
CONNECT_STYLE_CHANGE	드래그하여 연결시 그룹을 건너뛸때 스타일 변경 여부	Boolean
DELETABLE	가이드에 삭제 컨트롤러 여부	Boolean
LABEL_EDITABLE	라벨 수정여부	Boolean

위의 프로퍼티 값을 이용해서 몇가지 이벤트를 막아놓은 RectShape 도형을 그리는 예제입니다.

```
var canvas = new OG.Canvas('canvas', [1000, 800], 'white',
'url(resources/images/symbol/grid.gif');

//Block Edit Shape 만들기
var shape = new OG.RectangleShape('Block Edit');

//이동 불가
shape.MOVABLE = false;

//리사이징 불가
shape.RESIZABLE = false;

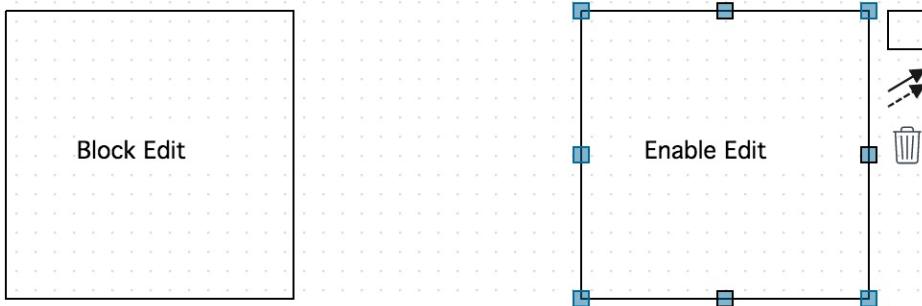
//연결 불가
shape.CONNECTABLE = false;

//삭제 불가
shape.DELETABLE = false;

//자기 자신 복사 불가
shape.CONNECT_CLONEABLE = false;

canvas.drawShape([200, 200], shape, [150, 150]);

canvas.drawShape([500, 200], new OG.RectangleShape('Enable Edit'), [150, 150]);
```



오픈그래프에서 제공하고 있는 IShape 를 확장한 5가지 기본 Shape 를 알아보도록 하겠습니다.

## GeomShape

OG.shape.G geomShape 클래스는 geometry 를 사용하여 드로잉하는 Shape 의 기본 클래스입니다.

OG.shape.G geomShape 를 바로 상속받아 사용하기 위해서는, 아래 예제와 같이 SHAPE\_ID 를 부여해주도록 합니다.

```

/**
 * Geometry Shape
 *
 * @class
 * @extends OG.shape.IShape
 * @requires OG.common.*
 * @requires OG.geometry.*
 *
 * @author <a href="mailto:sppark@uengine.org">Seungpil Park</a>
 */

```

Example)

```

var shape = new OG.GeoShape();
shape.SHPE_ID = 'OG.shape.CircleExample';
shape.createShape = function () {
    if (this.geom) {
        return this.geom;
    }
    this.geom = new OG.geometry.Circle([50, 50], 100);

    this.geom.style = new OG.geometry.Style({
        'cursor': 'default',
        'stroke': '#d9534f',
        'stroke-width': '1',
        'fill': 'none',
        'fill-opacity': 0
    });
    return this.geom;
};
var customShape = canvas.drawShape([300, 200], shape, [200, 200]);

```

## EdgeShape

OG.shape.EdgeShape 클래스는 선분 및 도형간의 선연결을 할 수 있는 Shape 입니다.

```

/**
 * Edge Shape
 *
 * @class
 * @extends OG.shape.IShape
 * @requires OG.common.*
 * @requires OG.geometry.*
 *
 * @param {Number[]} from 와이어 시작 좌표
 * @param {Number[]} to 와이어 끝 좌표
 * @param {String} label 라벨 [Optional]
 * @param {String} fromLabel 시작점 라벨 [Optional]
 * @param {String} toLabel 끝점 라벨 [Optional]
 * @author <a href="mailto:sppark@uengine.org">Seungpil Park</a>
 */

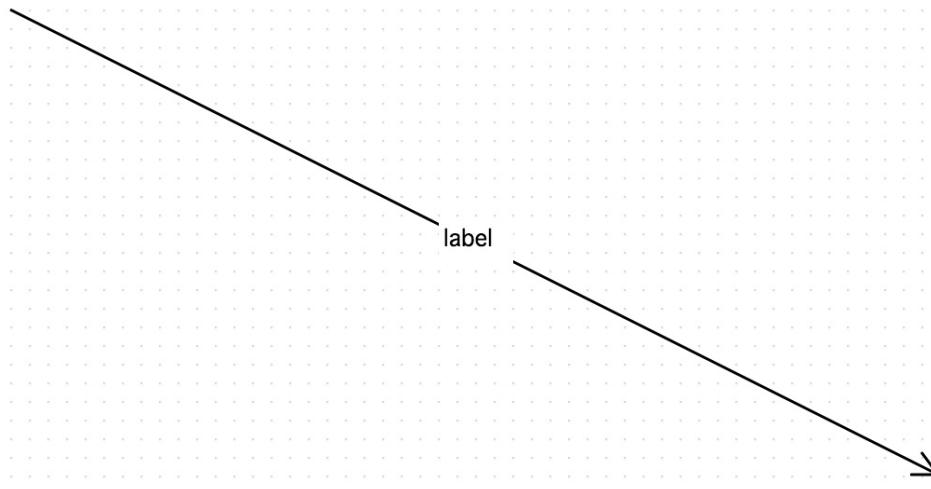
```

Example)

```

var canvas = new OG.Canvas('canvas', [1000, 800], 'white',
'url(resources/images/symbol/grid.gif)');
var edgeShape = new OG.shape.EdgeShape([100, 200], [600, 450], 'label');
canvas.drawShape(null, edgeShape, null, {'edge-type': 'plain', "arrow-start": "none", "arrow-end": "open-wide-long"});

```



도형끼리의 선 연결을 수행했을 경우 리턴되는 것도 EdgeShape 인데, 이렇게 생성된 연결 도형들은 Dom 객체의 attribute에 특정한 값을 가지게 됩니다.

다음 예제를 실행해봅니다.

```

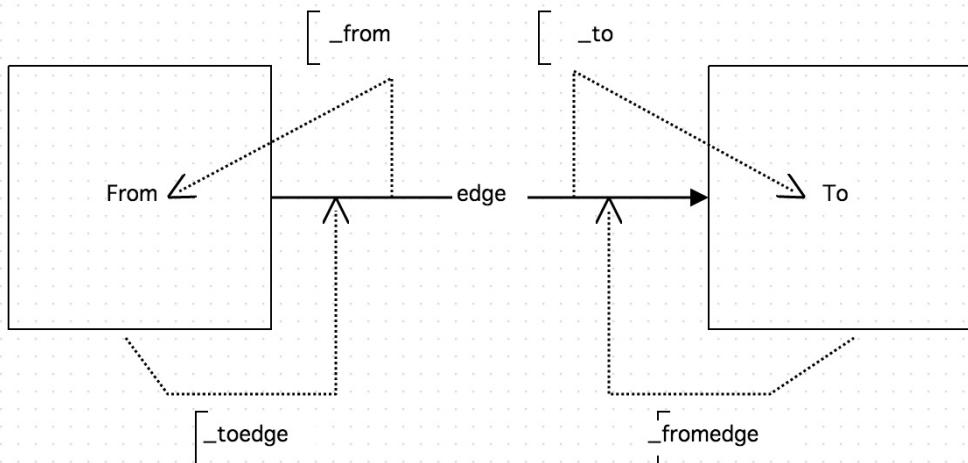
//From 도형
var element1 = canvas.drawShape([200,200],new OG.RectangleShape('From'),
[150,150]);

//To 도형
var element2 = canvas.drawShape([600,200],new OG.RectangleShape('To'),
[150,150]);

//connect 시킬 경우 edgeShape Dom 객체가 리턴된다.
var edge = canvas.connect(element1, element2, null, 'edge');

//각 어트리뷰트 확인
console.log($(element1).attr('_toedge'));
console.log($(element2).attr('_fromedge'));
console.log($(edge).attr('_from'));
console.log($(edge).attr('_to'));

```



Dom 객체 어트리뷰트	값 예시	설명	멀티 value
From element _toedge	OG_1277_8	자신으로 부터 외부로 나간 edge 아이디	콤마 세퍼레이터
To element _fromedge	OG_1277_8	자신을 향해 안으로 들어온 edge 아이디	콤마 세퍼레이터
Edge element _from	OG_1277_2_TERMINAL_50_50	선분과 from 으로 연결된 도형의 아이디 와 그 도형의 접점의 퍼센테이지 포인트	단일값
_to	OG_1277_5_TERMINAL_50_50	선분과 to 으로 연결된 도형의 아이디 와 그 도형의 접점의 퍼센테이지 포인트	단일값

이번에는, 화면에 이미 그려진 Edge 와 두 두형을 연결하는 예제를 실행해보겠습니다.

이미 그려진 Edge 를 연결하기 위해서는

`canvas.getRenderer().connect(fromTerminal,toTerminal,edgeElement)` 메소드를 사용합니다.

`fromTerminal` 과 `toTerminal` 은 Edge 의 `_from` 어트리뷰트와 `_to` 어트리뷰터 값입니다.

`terminal` 스트링을 만드는 것은 위의 표처럼 직접 스트링을 만드실 수도 있고,

`canvas.getRenderer().createDefaultTerminalString(fromElement|toElement)` 를 호출할 수 도 있습니다.

```

var fromElement = canvas.drawShape([200, 200], new OG.RectangleShape('From'),
[150, 150]);
var toElement = canvas.drawShape([600, 200], new OG.RectangleShape('To'),
[150, 150]);

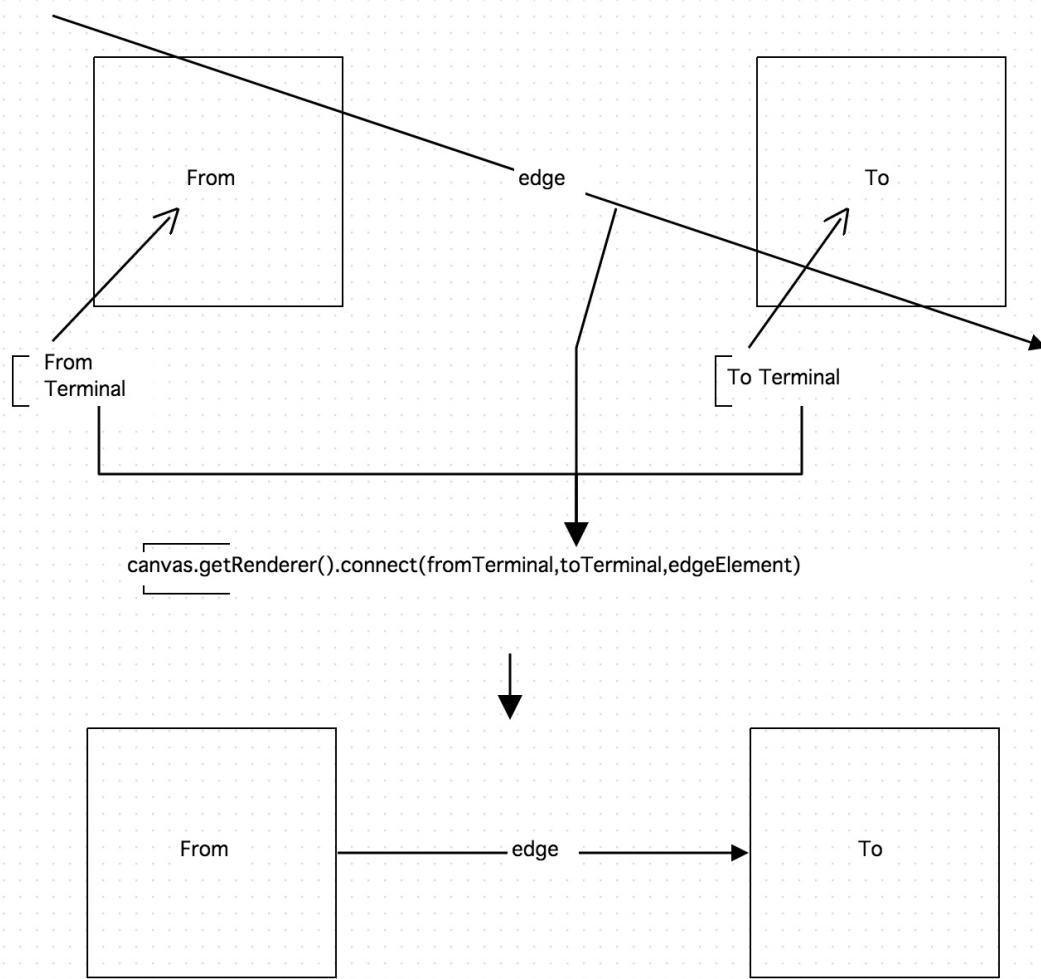
var edgeShape = new OG.shape.EdgeShape([100, 100], [700, 300], 'edge');
var edgeElement = canvas.drawShape(null, edgeShape);

//fromTerminal 스트링 만들기
var fromTerminal =
canvas.getRenderer().createDefaultTerminalString(fromElement);

//toTerminal 스트링 만들기
var toTerminal = canvas.getRenderer().createDefaultTerminalString(toElement);

//fromTerminal,toTerminal,edgeElement 를 연결하기
canvas.getRenderer().connect(fromTerminal,toTerminal,edgeElement);

```



## GroupShape

OG.shape.GroupShape 클래스는 기본적으로 OG.geometry.Rectangle 을 그리게 되며, 자신 안에 다른 도형을 포함시킬 수 있습니다.

앞서 [Canvas Drawing – group \(document/canvas.md#group\)](#) 파트에서 canvas.group([도형1,도형2...]) 메소드로 그룹핑을 할때 리턴되는 Dom 객체가 OG.shape.GroupShape 클래스를 shape 으로 가지고 있습니다.

```
/**  
 * Group Shape  
 *  
 * @class  
 * @extends OG.shape.IShape  
 * @requires OG.common.*  
 * @requires OG.geometry.*  
 *  
 * @param {String} label 라벨 [Optional]  
 * @author <a href="mailto:sppark@uengine.org">Seungpil Park</a>  
 */
```

Example)

```
var canvas = new OG.Canvas('canvas', [1000, 800], 'white',  
'url(resources/images/symbol/grid.gif');
```

```
var rectangleShape = canvas.drawShape([100, 100], new OG.RectangleShape('1'),  
[100, 100]);  
var circleShape = canvas.drawShape([300, 200], new OG.RectangleShape('2'), [80,  
80]);  
  
//create OG.shape.GroupShape Dom Element  
var group = canvas.group([rectangleShape,circleShape]);  
  
var addShape1 = canvas.drawShape([300, 100], new OG.CircleShape('add1'), [50,  
50]);  
var addShape2 = canvas.drawShape([100, 200], new OG.CircleShape('add2'), [50,  
50]);  
canvas.addToGroup(group,[addShape1,addShape2]);
```

OG.geometry.Rectangle 사각형 틀 이외의 다른 특수한 모양들도 GroupShape 로 만들 수 있습니다.

다음의 예제를 진행하여 봅니다.

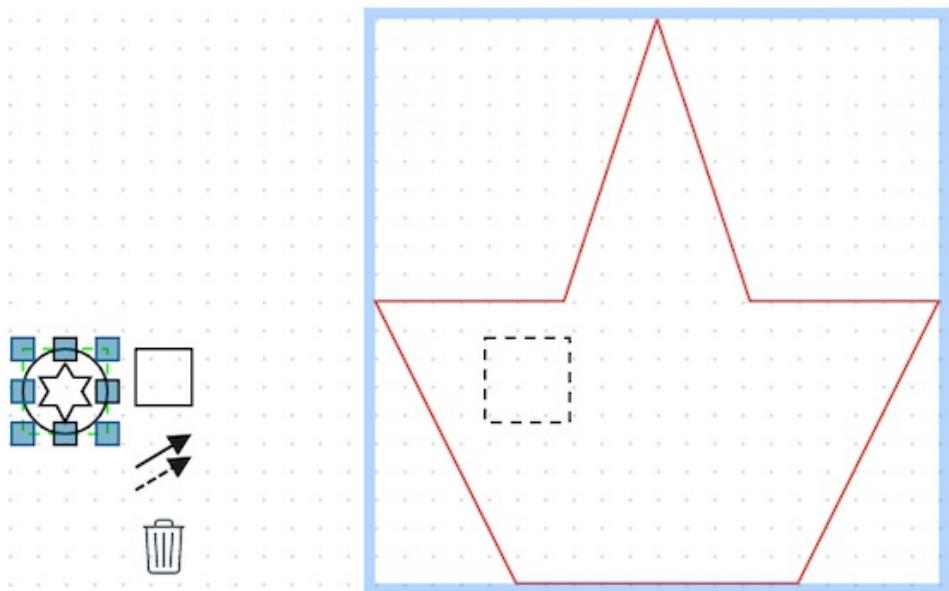
```

var shape = new OG.GroupShape();
shape.createShape = function () {
    if (this.geom) {
        return this.geom;
    }
    this.geom = new OG.geometry.PolyLine([[0,0],[50,50],[150,50],[75,100],
    [-75,100],[-150,50],[-50,50],[0,0]]);

    this.geom.style = new OG.geometry.Style({
        'cursor': 'default',
        'stroke': '#d9534f',
        'stroke-width': '1',
        'fill': 'none',
        'fill-opacity': 0
    });
    return this.geom;
};
var customShape = canvas.drawShape([300, 200], shape, [200, 200]);

```

위의 코드로 생성된 도형은 다른 도형을 자식으로 가질 수 있게 됩니다. 또한 UI에서 사용자가 드래그하여 다른 도형을 집어넣을 수 있습니다.



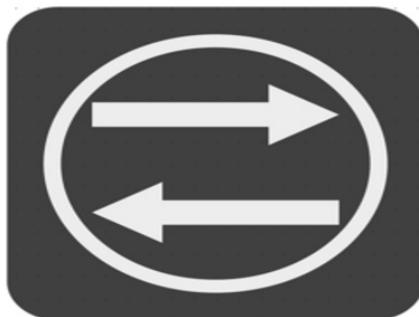
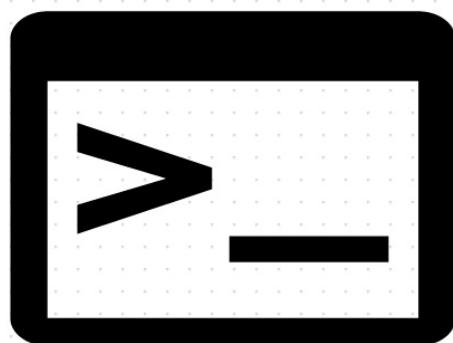
## ImageShape

OG.shape.ImageShape는 이미지를 드로잉 할 수 있는 shape 클래스입니다.

new OG.ImageShape(이미지 url)로 사용할 수 있습니다.

```
var imageShape1 = new OG.ImageShape('resources/images/router/Api.png');
var imageShape2 = new OG.ImageShape('resources/images/router/Function.svg');
var imageShape3 = new OG.ImageShape('resources/images/router/Oauth.png');
var imageShape4 = new OG.ImageShape('resources/images/router/Proxy.png');
var imageShape5 = new OG.ImageShape('resources/images/router/Request.png');
var imageShape6 = new OG.ImageShape('resources/images/router/Response.png');

canvas.drawShape([100, 100], imageShape1, [200, 150]);
canvas.drawShape([300, 100], imageShape2, [200, 150]);
canvas.drawShape([500, 100], imageShape3, [100, 100]);
canvas.drawShape([100, 300], imageShape4, [200, 150]);
canvas.drawShape([300, 300], imageShape5, [100, 100]);
canvas.drawShape([500, 300], imageShape6, [100, 100]);
```



## TextShape

OG.shape.TextShape 는 텍스트를 드로잉 할 수 있는 shape 클래스입니다.

```
var textShape1 = new OG.TextShape("Text Here, Hello World!");
var textShape2 = new OG.TextShape("Text Here, Hello World!");
var textShape3 = new OG.TextShape("Text Here, Hello World!");
var textShape4 = new OG.TextShape("Text Here, Hello World!");
var textShape5 = new OG.TextShape("Text Here, Hello World!");

canvas.drawShape([300, 100], textShape1, [200, 100, 45], {'font-size': 10,
'text-anchor': 'start', 'vertical-align': 'middle'});
canvas.drawShape([300, 150], textShape2, [200, 100], {'font-size': 15, 'text-
anchor': 'middle', 'vertical-align': 'top'});
canvas.drawShape([300, 200], textShape3, [200, 100], {'font-size': 20, 'text-
anchor': 'end', 'vertical-align': 'top'});
canvas.drawShape([300, 250], textShape4, [200, 100], {'font-size': 25, 'text-
anchor': 'middle', 'vertical-align': 'bottom'});
canvas.drawShape([300, 300], textShape5, [200, 100], {'font-size': 30, 'text-
anchor': 'end', 'vertical-align': 'bottom'});
```

Text Here, Hello World!

# Extend Shape

- [CircleShape](#)
- [EllipseShape](#)
- [RectangleShape](#)
- [PoolShape](#)
- [LaneShape](#)
- [Define Custom Shape](#)

[Shape \(shapes.md\)](#) 섹션에서 기본적인 다음의 기본적인 Shape 클래스들을 살펴보았습니다.

- [GeomShape \(shapes.md#geomshape\)](#)
- [EdgeShape \(shapes.md#edgeshape\)](#)
- [GroupShape \(shapes.md#groupshape\)](#)
- [ImageShape \(shapes.md#imageshape\)](#)
- [TextShape \(shapes.md#textshape\)](#)

다음 살펴볼 Shape 들은 위의 기본 Shape 클래스들을 상속받는 도형들입니다.

## CircleShape

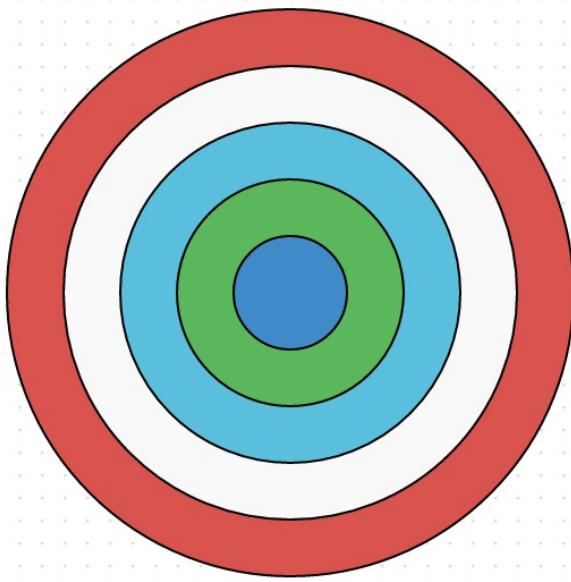
OG.shape.CircleShape 는 원형 도형입니다.

OG.shape.GeoShape 를 상속받습니다.

```
/**  
 * Circle Shape  
 *  
 * @class  
 * @extends OG.shape.GeoShape  
 * @requires OG.common.*  
 * @requires OG.geometry.*  
 *  
 * @param {String} label 라벨 [Optional]  
 * @author <a href="mailto:sppark@uengine.org">Seungpil Park</a>  
 */
```

Example)

```
canvas.drawShape([300, 300], new OG.CircleShape(), [250, 250], {'fill': '#d9534f', 'fill-opacity': 1});  
canvas.drawShape([300, 300], new OG.CircleShape(), [200, 200], {'fill': '#f9f9f9', 'fill-opacity': 1});  
canvas.drawShape([300, 300], new OG.CircleShape(), [150, 150], {'fill': '#5bc0de', 'fill-opacity': 1});  
canvas.drawShape([300, 300], new OG.CircleShape(), [100, 100], {'fill': '#5cb85c', 'fill-opacity': 1});  
canvas.drawShape([300, 300], new OG.CircleShape(), [50, 50], {'fill': '#428bca', 'fill-opacity': 1});
```



## EllipseShape

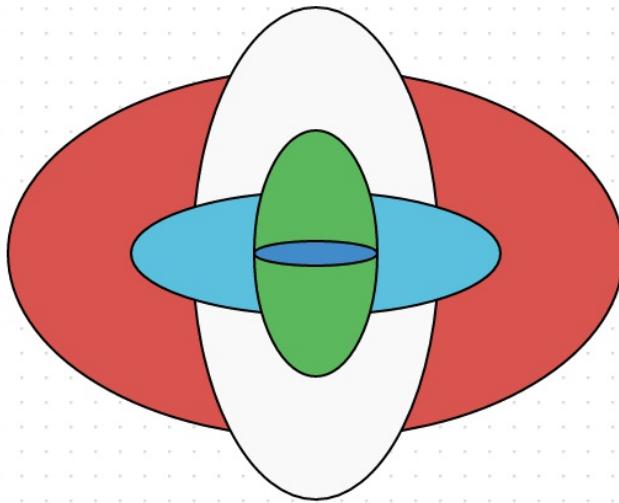
OG.shape.EllipseShape 는 타원형 도형입니다.

OG.shape.GeomShape 를 상속받습니다.

```
/**  
 * Ellipse Shape  
 *  
 * @class  
 * @extends OG.shape.GeomShape  
 * @requires OG.common.*  
 * @requires OG.geometry.*  
 *  
 * @param {String} label 라벨 [Optional]  
 * @author <a href="mailto:sppark@uengine.org">Seungpil Park</a>  
 */
```

Example)

```
canvas.drawShape([300, 300], new OG.EllipseShape(), [250, 150], {'fill':  
  '#d9534f', 'fill-opacity': 1});  
canvas.drawShape([300, 300], new OG.EllipseShape(), [100, 200], {'fill':  
  '#f9f9f9', 'fill-opacity': 1});  
canvas.drawShape([300, 300], new OG.EllipseShape(), [150, 50], {'fill':  
  '#5bc0de', 'fill-opacity': 1});  
canvas.drawShape([300, 300], new OG.EllipseShape(), [50, 100], {'fill':  
  '#5cb85c', 'fill-opacity': 1});  
canvas.drawShape([300, 300], new OG.EllipseShape(), [50, 10], {'fill':  
  '#428bca', 'fill-opacity': 1});
```



## RectangleShape

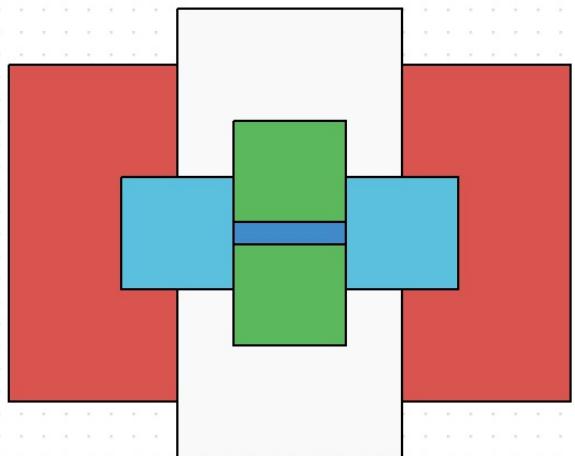
OG.shape.EllipseShape 는 직사각형 도형입니다.

OG.shape.GeoShape 를 상속받습니다.

```
/**  
 * Rectangle Shape  
 *  
 * @class  
 * @extends OG.shape.GeoShape  
 * @requires OG.common.*  
 * @requires OG.geometry.*  
 *  
 * @param {String} label 라벨 [Optional]  
 * @author <a href="mailto:sppark@uengine.org">Seungpil Park</a>  
 */
```

Example)

```
canvas.drawShape([300, 300], new OG.RectangleShape(), [250, 150], {'fill':  
  '#d9534f', 'fill-opacity': 1});  
canvas.drawShape([300, 300], new OG.RectangleShape(), [100, 200], {'fill':  
  '#f9f9f9', 'fill-opacity': 1});  
canvas.drawShape([300, 300], new OG.RectangleShape(), [150, 50], {'fill':  
  '#5bc0de', 'fill-opacity': 1});  
canvas.drawShape([300, 300], new OG.RectangleShape(), [50, 100], {'fill':  
  '#5cb85c', 'fill-opacity': 1});  
canvas.drawShape([300, 300], new OG.RectangleShape(), [50, 10], {'fill':  
  '#428bca', 'fill-opacity': 1});
```



## PoolShape

PoolShape 는 OG.shape.GroupShape 를 상속받으며 Label 이 도형의 상단라인에 위치하는 특성이 있습니다.

PoolShape 의 종류는 OG.shape.VerticalPoolShape 와 OG.shape.HorizontalPoolShape 이 있습니다.

PoolShape 의 자식으로 그려진 도형은, PoolShape 외부에 그려진 도형과 선연결이 될 경우 연결 스타일이 바뀌게 되는 속성이 있는데, 이는 BPM 의 일반적 표현입니다.

이를 방지 하기 위해서는 canvas 의 CONNECT\_STYLE\_CHANGE 컨피그를 false 로 주시면 됩니다.

```

var verticalPool = canvas.drawShape([200, 200], new
OG.VerticalPoolShape('VerticalPoolShape'), [250, 200], {
  'fill': '#d9534f',
  'fill-opacity': 1
});
var horizontalPool = canvas.drawShape([500, 200], new
OG.HorizontalPoolShape('HorizontalPoolShape'), [250, 200], {
  'fill': '#428bca',
  'fill-opacity': 1
});

//verticalPool 안에 도형을 그린다.
var startShape1 = canvas.drawShape([250, 150], new OG.E_Start(), [30, 30]);
canvas.appendChild(startShape1, verticalPool);

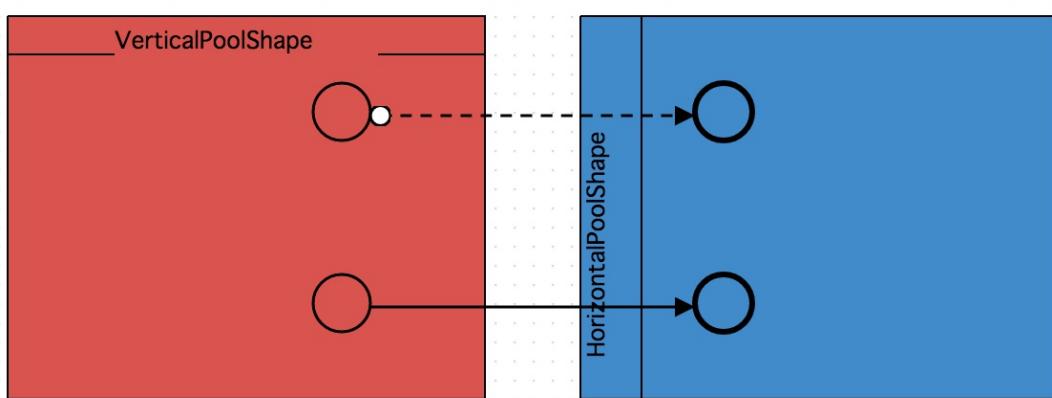
//horizontalPool 안에 도형을 그린다.
var endShape1 = canvas.drawShape([450, 150], new OG.E_End(), [30, 30]);
canvas.appendChild(endShape1, horizontalPool);

//startShape1 과 endShape1 를 연결한다.(Group 간의 선 연결)
canvas.connect(startShape1, endShape1);

//Group 간의 연결시 선 스타일의 변경 여부를 false 로 줄 경우
canvas._CONFIG.CONNECT_STYLE_CHANGE = false;
var startShap2 = canvas.drawShape([250, 250], new OG.E_Start(), [30, 30]);
canvas.appendChild(startShap2, verticalPool);

var endShape2 = canvas.drawShape([450, 250], new OG.E_End(), [30, 30]);
canvas.appendChild(endShape2, horizontalPool);
canvas.connect(startShap2, endShape2);

```



## LaneShape

LaneShape 는 PoolShape 와 마찬가지로 OG.shape.GroupShape 를 상속받으며 Label 이 도형의 상단라인에 위치하는 특성이 있고, UI 상에서 도형에 대한 분기를 실행할 수 있습니다.

LaneShape 의 종류는 OG.shape.VerticalLaneShape 와 OG.shape.HorizontalLaneShape 이 있습니다.

PoolShape 와 마찬가지로 내부의 도형이 외부의 도형과 연결될 경우 선연결의 변화를 방지하려면 CONNECT\_STYLE\_CHANGE 컨피그를 false 로 주시면 됩니다.

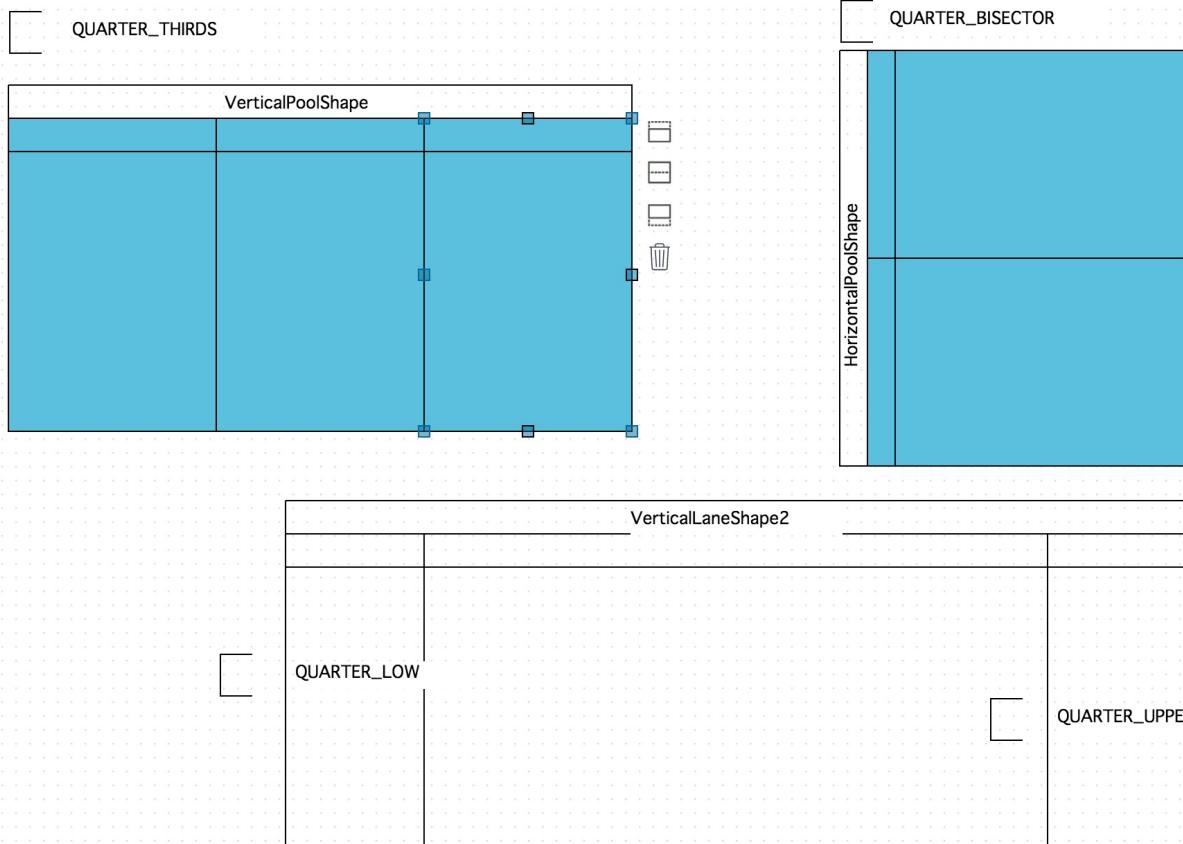
UI 이외에 프로그램 적으로 LaneShape 를 분기할 때는, canvas.getRenderer().divideLane(LaneShape Dom element, 분기명령) 으로 실행합니다.

분기 명령	방향
QUARTER_UPPER	VerticalLane 인 경우 좌측 방향, Horizontal 인 경우 상단 방향으로 확장
QUARTER_LOW	VerticalLane 인 경우 우측 방향, Horizontal 인 경우 하단 방향으로 확장
QUARTER_BISECTOR	내부적으로 2분기
QUARTER_THIRDS	내부적으로 3분기

```
var verticalLane = canvas.drawShape([300, 300], new
OG.VerticalLaneShape('VerticalLaneShape'), [450, 250]);
var divideLanes = canvas.getRenderer().divideLane(verticalLane,
'QUARTER_THIRDS');
for(var i = 0 ; i <divideLanes.length; i++){
    canvas.setShapeStyle(divideLanes[i], {'fill':'#5bc0de', 'fill-opacity':1});
}

var horizontalLane = canvas.drawShape([800, 300], new
OG.HorizontalLaneShape('HorizontalLaneShape'), [250, 300]);
divideLanes =
canvas.getRenderer().divideLane(horizontalLane, 'QUARTER_BISECTOR');
for(var i = 0 ; i <divideLanes.length; i++){
    canvas.setShapeStyle(divideLanes[i], {'fill':'#5bc0de', 'fill-opacity':1});
}

var verticalLane2 = canvas.drawShape([600, 600], new
OG.VerticalLaneShape('VerticalLaneShape2'), [450, 250]);
canvas.getRenderer().divideLane(verticalLane2, 'QUARTER_LOW');
canvas.getRenderer().divideLane(verticalLane2, 'QUARTER_UPPER');
```



## Define Custom Shape

OG.shape 네임스페이스에 새로운 사용자 지정 Shape 를 등록하여 사용하는 방법을 알아보겠습니다.

구현을 위해 [GeometryCollection \(geometry.md#geometrycollection\)](#) 의 예제를 재사용하도록 하겠습니다.

```
var shape = new OG.GeoShape();
shape.SHAPE_ID = 'OG.shape.CustomCollection';
shape.createShape = function () {
.
.
```

예제에서는 GeoShape 클래스를 하나 생성하고, OG.shape.CustomCollection 아이디를 부여한 후, createShape 메소드를 오버라이딩 하여 사용했습니다.

이번 파트에서 진행할 내용은 오픈그래프 네임스페이스에 OG.shape.CustomCollection 를 등록하고, 필요할 경우 이 Shape 을 재사용할 수 있도록 하는 예제입니다.

Html 페이지가 시작할 때, 오픈그래프 라이브러리를 읽어들인 후, 다음의 스크립트를 실행하도록 하십시오.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ko" lang="ko">
<head>
    <title>BPMN Modeler Example</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
    <meta http-equiv="X-UA-Compatible" content="IE=EmulateIE8">
```

```

<!-- jquery -->
<script type="text/javascript" src="./lib/jquery-1.11.1/jquery-
1.11.1.min.js"></script>

<!-- jquery ui -->
<script type="text/javascript" src="./lib/jquery-ui-1.11.0.custom/jquery-
ui.min.js"></script>
<link rel="stylesheet" type="text/css" href="./lib/jquery-ui-
1.11.0.custom/jquery-ui.css"/>

<!-- jquery Context Menu -->
<link rel="stylesheet" type="text/css"
href="./lib/contextmenu/jquery.contextMenu.css"/>
<script type="text/javascript" src="./lib/contextmenu/jquery.contextMenu-
min.js"></script>

<!-- Opengraph -->
<script type="text/javascript" src="./lib/opengraph/OpenGraph-0.1.1-
SNAPSHOT.js"></script>

<script type="text/javascript">
$(document).ready(function () {
    OG.shape.CustomCollection = function (label) {
        OG.shape.CustomCollection.superclass.call(this);

        this.SHAPE_ID = 'OG.shape.CustomCollection';
        this.label = label;
    };
    OG.shape.CustomCollection.prototype = new OG.shape.GeoShape();
    OG.shape.CustomCollection.superclass = OG.shape.GeoShape;
    OG.shape.CustomCollection.prototype.constructor =
OG.shape.CustomCollection;
    OG.CustomCollection = OG.shape.CustomCollection;

});

</script>
</head>
<body>

<div id="canvas" style="cursor: default;"></div>

</body>
</html>

```

위의 코드에서는 OG.shape.GeoShape() 클래스를 상속받은 OG.shape.CustomCollection 클래스를 생성하였고, SHAPE\_ID 로 'OG.shape.CustomCollection' 를 지정하였습니다.

이제 이 클래스가 도형을 어떻게 그릴 것인가 정의하는 createShape 메소드를 생성할 차례입니다.

```

$(document).ready(function () {
    OG.shape.CustomCollection = function (label) {
        OG.shape.CustomCollection.superclass.call(this);

        this.SHAPE_ID = 'OG.shape.CustomCollection';
        this.label = label;
    };
    OG.shape.CustomCollection.prototype = new OG.shape.GemShape();
    OG.shape.CustomCollection.superclass = OG.shape.GemShape;
    OG.shape.CustomCollection.prototype.constructor =
    OG.shape.CustomCollection;
    OG.CustomCollection = OG.shape.CustomCollection;

    OG.shape.CustomCollection.prototype.createShape = function () {
        if (this.geom) {
            return this.geom;
        }

        var geomCollection = [];
        if (this.geom) {
            return this.geom;
        }

        for (var i = 0; i < 36; i++) {
            var geom = new OG.geometry.Curve([[500, 200], [400, 400 - i * 2],
[200, 0 + i * 2], [100, 200]]);
            geomCollection.push(geom);
        }
        for (var i = 0; i < 15; i++) {
            geomCollection.push(new OG.geometry.Circle([300, 200], i * 2));
        }

        this.geom = new OG.geometry.GeometryCollection(geomCollection);

        this.geom.style = new OG.geometry.Style({
            'cursor': 'default',
            'stroke': '#d9534f',
            'stroke-width': '1',
            'fill': 'none',
            'fill-opacity': 0
        });
        return this.geom;
    };
});

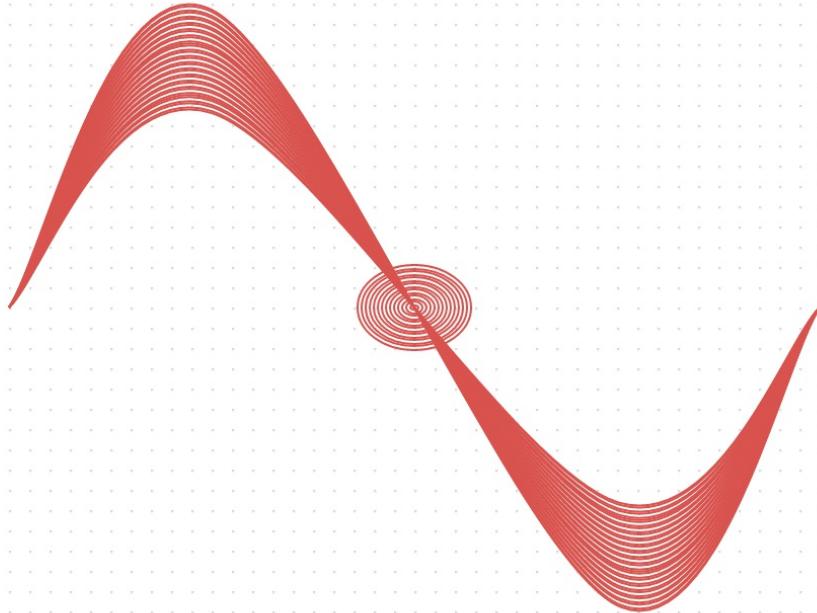
```

이제 오픈그래프 네임스페이스에 OG.shape.CustomCollection 도형을 그릴 준비가 완료되었습니다.

실제로 이 클래스를 사용하여 화면에 드로잉 해 보도록 합니다.

```
var canvas = new OG.Canvas('canvas', [1000, 800], 'white',
'url(resources/images/symbol/grid.gif');

var shape = new OG.CustomCollection();
var customShape = canvas.drawShape([400, 300], shape, [400, 300]);
```



OG.shape.CustomCollection 에 특별한 메소드를 만들어보도록 합니다.

createShape 메소드 바로 아래에, animation 이라는 새로운 프로포타입 메소드를 생성하도록 합니다.

```

.
.

OG.shape.CustomCollection.prototype.animation = function(canvas, element){
    var count = 0;
    var command = 'plus';
    var interval = setInterval(function(){
        if(command == 'plus' && count > 10){
            command = 'minus';
        }else if(command == 'minus' && count < 1){
            command = 'plus';
        }
        if(command == 'plus'){
            count++;
        }else{
            count--;
        }
        var geomCollection = [];
        for (var i = 0; i < 36; i++) {
            var geom = new OG.geometry.Curve([[500, 200], [400, 400 - i * count], [200, 0 + i * count], [100, 200]]);
            geomCollection.push(geom);
        }
        for (var i = 0; i < 15; i++) {
            geomCollection.push(new OG.geometry.Circle([300, 200], i * count));
        }
        element.shape.geom = new
OG.geometry.GeometryCollection(geomCollection);
        element.shape.geom.style = new OG.geometry.Style({
            'cursor': 'default',
            'stroke': '#d9534f',
            'stroke-width': '1',
            'fill': 'none',
            'fill-opacity': 0
        });

        canvas.getRenderer().redrawShape(element);
    },100);
};

```

위의 animation 메소드는, canvas 와 Dom 객체를 인자값으로 받아, 0.1 초마다 변화하는 geometry 를 캔버스 상의 Dom 객체에 적용시킬 것입니다.

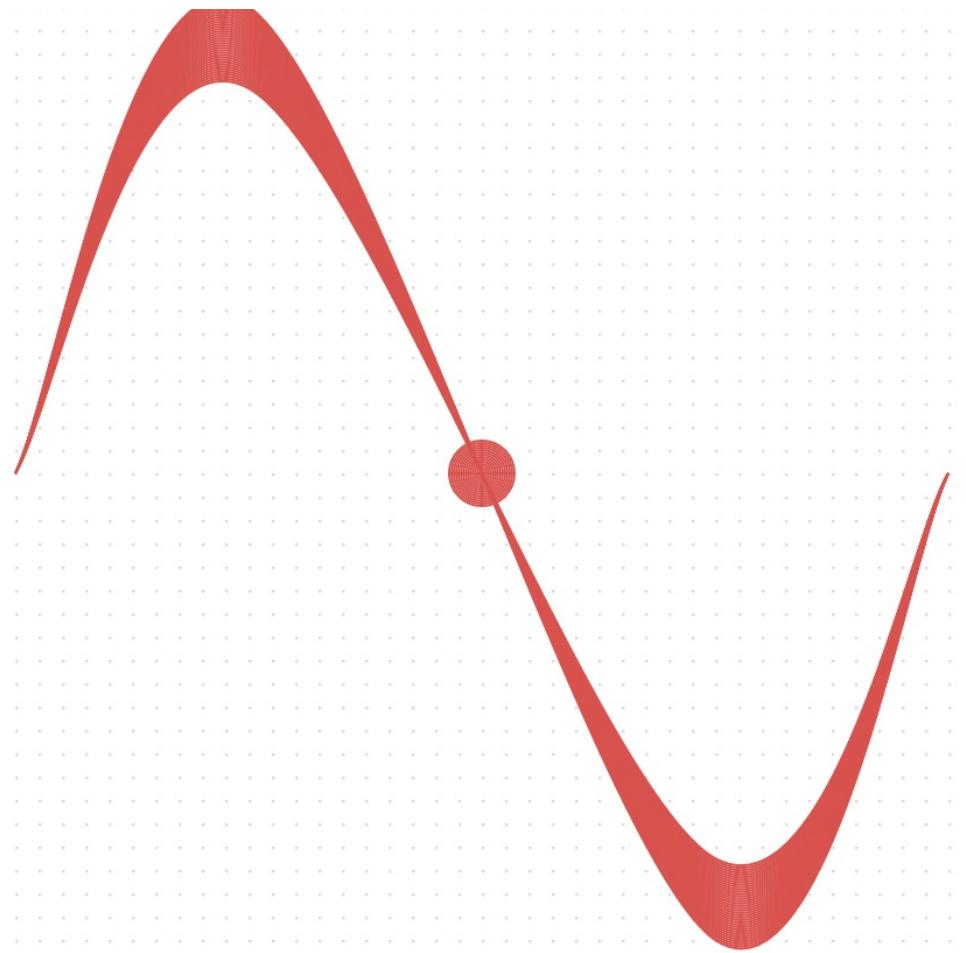
animation 메소드를 실행시켜 보도록 합니다.

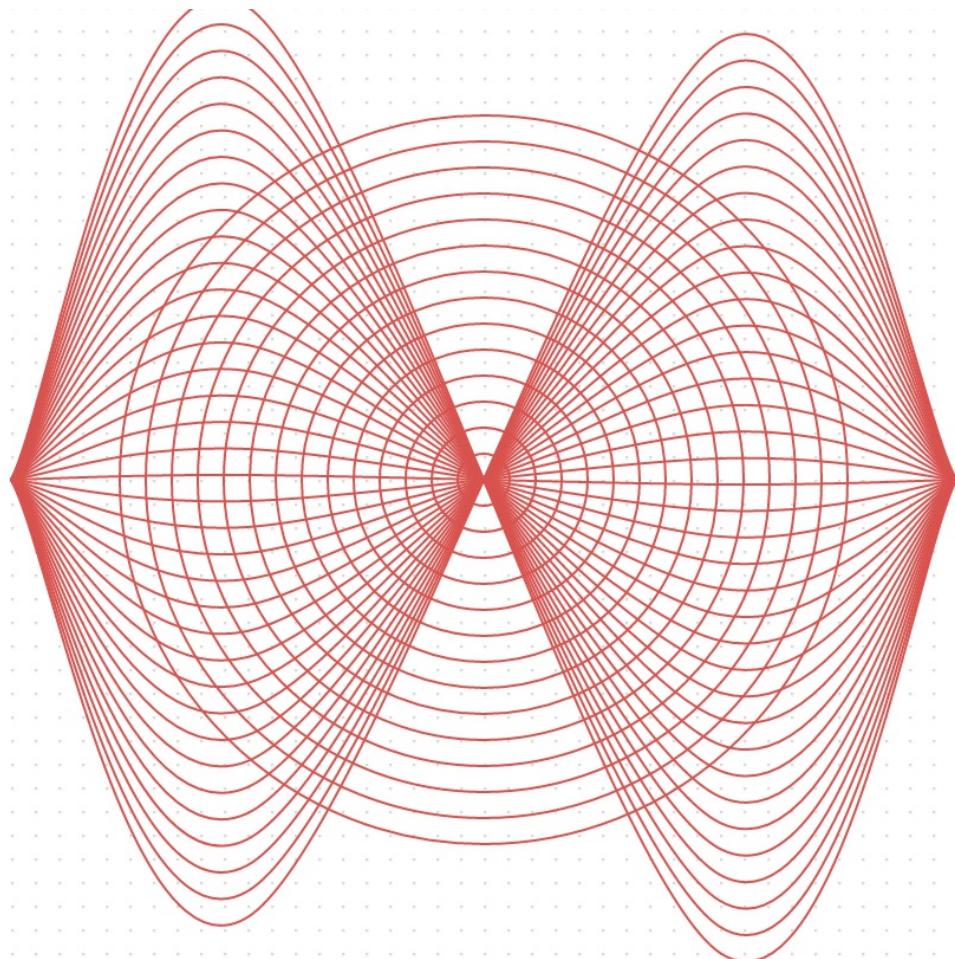
```

var canvas = new OG.Canvas('canvas', [1000, 800], 'white',
'url(resources/images/symbol/grid.gif');

var shape = new OG.CustomCollection();
var customShape = canvas.drawShape([400, 300], shape, [400, 300]);
customShape.shape.animation(canvas, customShape);

```





- 풀 소스 코드

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ko" lang="ko">
<head>
    <title>BPMN Modeler Example</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
    <meta http-equiv="X-UA-Compatible" content="IE=EmulateIE8">

    <!-- jquery -->
    <script type="text/javascript" src=".//lib/jquery-1.11.1/jquery-
1.11.1.min.js"></script>

    <!-- jquery ui -->
    <script type="text/javascript" src=".//lib/jquery-ui-1.11.0.custom/jquery-
ui.min.js"></script>
    <link rel="stylesheet" type="text/css" href=".//lib/jquery-ui-
1.11.0.custom/jquery-ui.css"/>

    <!-- jquery Context Menu -->
    <link rel="stylesheet" type="text/css"
href=".//lib/contextmenu/jquery.contextMenu.css"/>
    <script type="text/javascript" src=".//lib/contextmenu/jquery.contextMenu-
```

```
min.js"></script>

<!-- Opengraph -->
<script type="text/javascript" src="./lib/opengraph/OpenGraph-0.1.1-
SNAPSHOT.js"></script>

<script type="text/javascript">
$(document).ready(function () {
    OG.shape.CustomCollection = function (label) {
        OG.shape.CustomCollection.superclass.call(this);

        this.SHAPE_ID = 'OG.shape.CustomCollection';
        this.label = label;
    };
    OG.shape.CustomCollection.prototype = new OG.shape.GeoShape();
    OG.shape.CustomCollection.superclass = OG.shape.GeoShape;
    OG.shape.CustomCollection.prototype.constructor =
OG.shape.CustomCollection;
    OG.CustomCollection = OG.shape.CustomCollection;

    OG.shape.CustomCollection.prototype.createShape = function () {
        if (this.geom) {
            return this.geom;
        }

        var geomCollection = [];
        if (this.geom) {
            return this.geom;
        }

        for (var i = 0; i < 36; i++) {
            var geom = new OG.geometry.Curve([[500, 200], [400, 400 - i
* 2], [200, 0 + i * 2], [100, 200]]);
            geomCollection.push(geom);
        }
        for (var i = 0; i < 15; i++) {
            geomCollection.push(new OG.geometry.Circle([300, 200], i *
2));
        }
    }

    this.geom = new OG.geometry.GeometryCollection(geomCollection);

    this.geom.style = new OG.geometry.Style({
        'cursor': 'default',
        'stroke': '#d9534f',
        'stroke-width': '1',
        'fill': 'none',
        'fill-opacity': 0
    });
    return this.geom;
};


```

```

OG.shape.CustomCollection.prototype.animation = function(canvas,
element){
    var count = 0;
    var command = 'plus';
    var interval = setInterval(function(){
        if(command == 'plus' && count > 10){
            command = 'minus';
        }else if(command == 'minus' && count < 1){
            command = 'plus';
        }
        if(command == 'plus'){
            count++;
        }else{
            count--;
        }
        var geomCollection = [];
        for (var i = 0; i < 36; i++) {
            var geom = new OG.geometry.Curve([[500, 200], [400, 400
- i * count], [200, 0 + i * count], [100, 200]]);
            geomCollection.push(geom);
        }
        for (var i = 0; i < 15; i++) {
            geomCollection.push(new OG.geometry.Circle([300, 200],
i * count));
        }
        element.shape.geom = new
OG.geometry.GeometryCollection(geomCollection);
        element.shape.geom.style = new OG.geometry.Style({
            'cursor': 'default',
            'stroke': '#d9534f',
            'stroke-width': '1',
            'fill': 'none',
            'fill-opacity': 0
        });
        canvas.getRenderer().redrawShape(element);
    },100);
};

var canvas = new OG.Canvas('canvas', [1000, 800], 'white',
'url(resources/images/symbol/grid.gif');

var shape = new OG.CustomCollection();
var customShape = canvas.drawShape([400, 300], shape, [400, 300]);
customShape.shape.animation(canvas, customShape);

});

</script>
</head>
<body>
```

```
<div id="canvas" style="cursor: default;"></div>  
</body>  
</html>
```

# Marker

- [Default Marker](#)
- [Custom Marker](#)
- [Custom Marker Size](#)
- [Custom Marker Ref](#)
- [Custom Marker Style](#)

마커 스타일링은 [EdgeShape \(shapes.md#edgeshape\)](#) 에 적용되며, 선 양 끝 또는 중앙, 변곡점 구간에 SVG 마커를 표현합니다.

## [SVG Marker Element](#)

### Default Marker

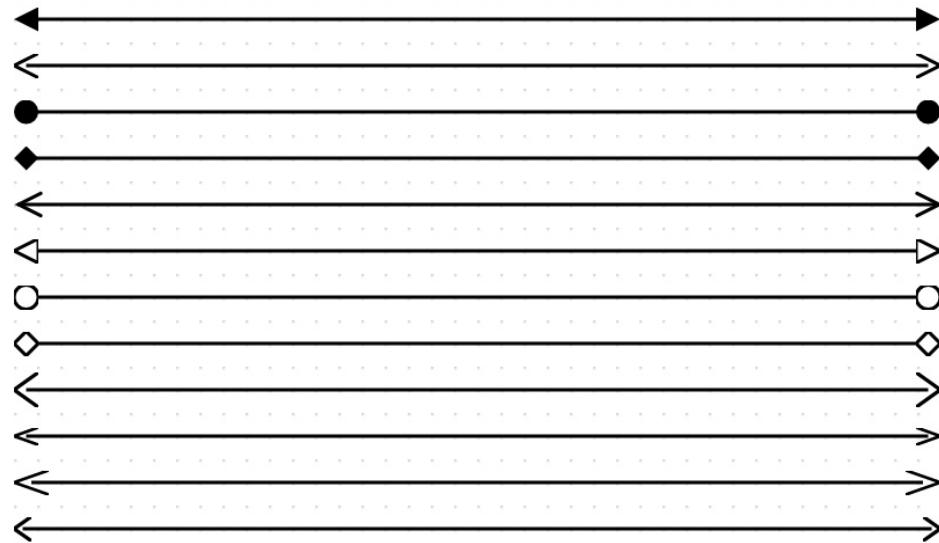
오픈그래프의 디폴트 마커는 arrow-start 와 arrow-end 스타일 어트리뷰트로 설정할 수 있으며, block,classic 등 12 가지의 모양을 가지고 있습니다.

스타일 어트리뷰트	설명
arrow-start	선 시작부분의 마커
arrow-end	선 끝부분의 마커

위의 어트리뷰트 값을 12가지의 기본 모형을 적용하여 샘플코드를 작성해 봅니다.

```
var edge1 = canvas.drawShape(null, new OG.EdgeShape([100, 100], [500, 100]), null);
var edge2 = canvas.drawShape(null, new OG.EdgeShape([100, 120], [500, 120]), null);
var edge3 = canvas.drawShape(null, new OG.EdgeShape([100, 140], [500, 140]), null);
var edge4 = canvas.drawShape(null, new OG.EdgeShape([100, 160], [500, 160]), null);
var edge5 = canvas.drawShape(null, new OG.EdgeShape([100, 180], [500, 180]), null);
var edge6 = canvas.drawShape(null, new OG.EdgeShape([100, 200], [500, 200]), null);
var edge7 = canvas.drawShape(null, new OG.EdgeShape([100, 220], [500, 220]), null);
var edge8 = canvas.drawShape(null, new OG.EdgeShape([100, 240], [500, 240]), null);
var edge9 = canvas.drawShape(null, new OG.EdgeShape([100, 260], [500, 260]), null);
var edge10 = canvas.drawShape(null, new OG.EdgeShape([100, 280], [500, 280]), null);
var edge11 = canvas.drawShape(null, new OG.EdgeShape([100, 300], [500, 300]), null);
var edge12 = canvas.drawShape(null, new OG.EdgeShape([100, 320], [500, 320]), null);

canvas.setShapeStyle(edge1, {'arrow-start': 'block', 'arrow-end': 'block'});
canvas.setShapeStyle(edge2, {'arrow-start': 'classic', 'arrow-end': 'classic'});
canvas.setShapeStyle(edge3, {'arrow-start': 'oval', 'arrow-end': 'oval'});
canvas.setShapeStyle(edge4, {'arrow-start': 'diamond', 'arrow-end': 'diamond'});
canvas.setShapeStyle(edge5, {'arrow-start': 'open', 'arrow-end': 'open'});
canvas.setShapeStyle(edge6, {'arrow-start': 'open_block', 'arrow-end': 'open_block'});
canvas.setShapeStyle(edge7, {'arrow-start': 'open_oval', 'arrow-end': 'open_oval'});
canvas.setShapeStyle(edge8, {'arrow-start': 'open_diamond', 'arrow-end': 'open_diamond'});
canvas.setShapeStyle(edge9, {'arrow-start': 'wide', 'arrow-end': 'wide'});
canvas.setShapeStyle(edge10, {'arrow-start': 'narrow', 'arrow-end': 'narrow'});
canvas.setShapeStyle(edge11, {'arrow-start': 'long', 'arrow-end': 'long'});
canvas.setShapeStyle(edge12, {'arrow-start': 'short', 'arrow-end': 'short'});
```



## Custom Marker

오픈그래프는 사용자 스스로 마커를 제작하여 스타일링에 활용할 수 있도록 지원합니다.

커스텀 마커를 사용하기 위해서는, 우선 마커 클래스 하나를 제작해보도록 합니다.

마커 클래스 제작은 [Define Custom Shape \(extend-shape.md#define-custom-shape\)](#) 제작방식과 동일하지만 네임스페이스만 바뀝니다.

다음은 사각형 안에 N 모양이 있는 커스텀 마커를 제작하는 샘플 코드입니다.

```
/**  
 * Rectangle Maker  
 *  
 * @class  
 * @extends OG.marker.IMarker  
 * @requires OG.common.*  
 * @requires OG.geometry.*  
 *  
 * @author <a href="mailto:sppark@uengine.org">Seungpil Park</a>  
 */  
OG.marker.RectangleMarker = function () {  
    OG.marker.RectangleMarker.superclass.call(this);  
  
    this.MARKER_ID = 'OG.marker.RectangleMarker';  
};  
OG.marker.RectangleMarker.prototype = new OG.marker.IMarker();  
OG.marker.RectangleMarker.superclass = OG.marker.IMarker;  
OG.marker.RectangleMarker.prototype.constructor = OG.marker.RectangleMarker;  
OG.RectangleMarker = OG.marker.RectangleMarker;  
  
/**  
 * 드로잉할 marker 을 생성하여 반환한다.  
 */
```

```

 * @return {OG.geometry.Geometry} marker 정보
 * @override
 */
OG.marker.RectangleMarker.prototype.createMarker = function () {
    var geom1, geom2, geomCollection = [];
    if (this.geom) {
        return this.geom;
    }

    geom1 = new OG.geometry.Circle([50, 50], 50);
    geom1.style = new OG.geometry.Style({
        "stroke-width": 4
    });

    geom2 = new OG.geometry.Polygon([
        [20, 75],
        [40, 30],
        [60, 60],
        [80, 20],
        [60, 75],
        [40, 50]
    ]);

    geom2.style = new OG.geometry.Style({
        "fill": "black",
        "fill-opacity": 1
    });

    geomCollection.push(geom1);
    geomCollection.push(geom2);

    this.geom = new OG.geometry.GeometryCollection(geomCollection);

    return this.geom;
};

```

## Custom Marker Size

커스텀 Shape 를 생성할때와 모든 코드가 동일하지만, OG.shape 가 OG.marker 로 바뀌었음을 알 수 있습니다.

이제 이 코드를 오픈그래프 라이브러리를 불러온 후 html 페이지에 삽입하도록 합니다.

다음 할 일은 이 마커를 선 도형에 적용시키는 것입니다.

선 도형에 적용시킬 때는 "marker" 스타일 어트리뷰트를 사용하게 되는데, 몇가지 프로퍼티와 함께 json 형식으로 표현하도록 합니다.

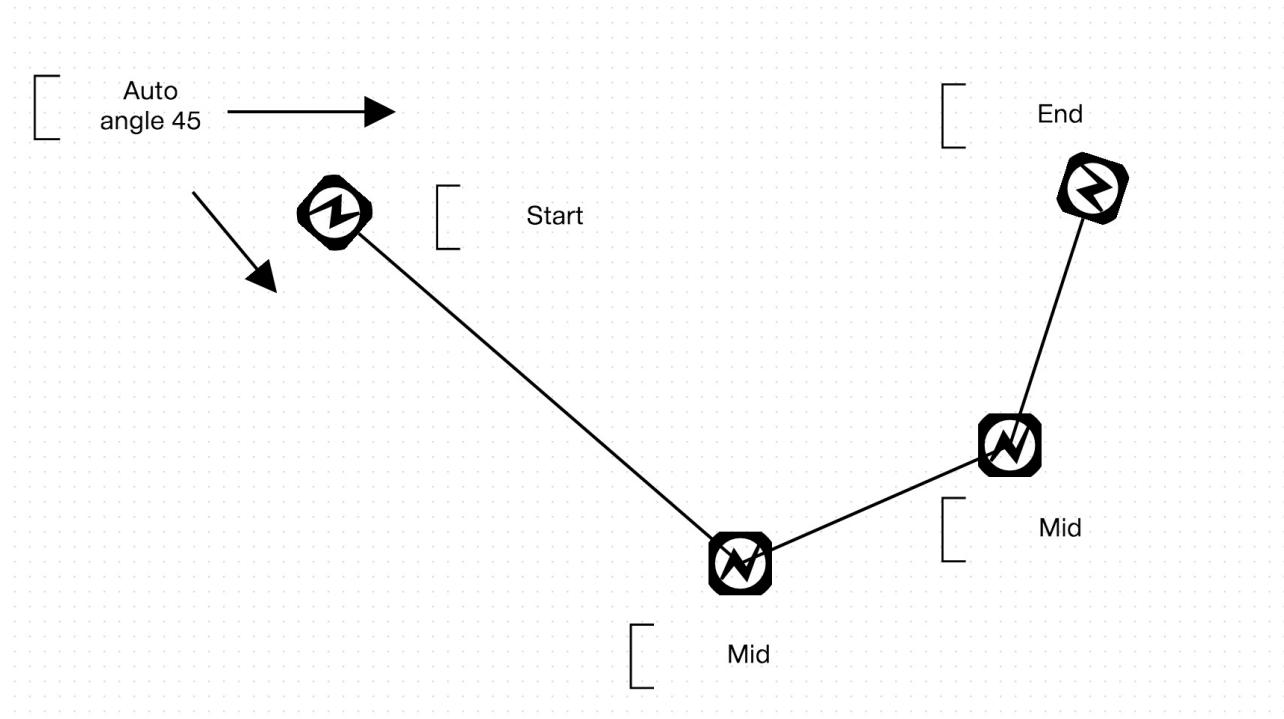
위치 프로퍼티	타입	설명
start	Object	선 시작부분의 마커
end	Object	선 끝부분의 마커
mid	Object	선 꺽임 부분의 마커
표현 프로퍼티	설명	설명

id	string	마커 클래스명
size	[number,number]	가로, 세로 크기
ref	[number,number]	경로상의 마커 참조 위치
style	Object	마커 스타일링

위의 표를 참조하여, 선의 시작, 꺽임, 끝 부분에 제작한 OG.marker.RectangleMarker 마커를 적용하도록 해봅니다.

```
var edge1 = canvas.drawShape(null, new OG.EdgeShape([100, 100], [500, 100]), null);

canvas.setShapeStyle(edge1, {
    'marker': {
        'start': {
            'id': 'OG.marker.RectangleMarker',
            'size': [20, 20]
        },
        'mid': {
            'id': 'OG.marker.RectangleMarker',
            'size': [20, 20]
        },
        'end': {
            'id': 'OG.marker.RectangleMarker',
            'size': [20, 20]
        }
    }
});
```



## Custom Marker Ref

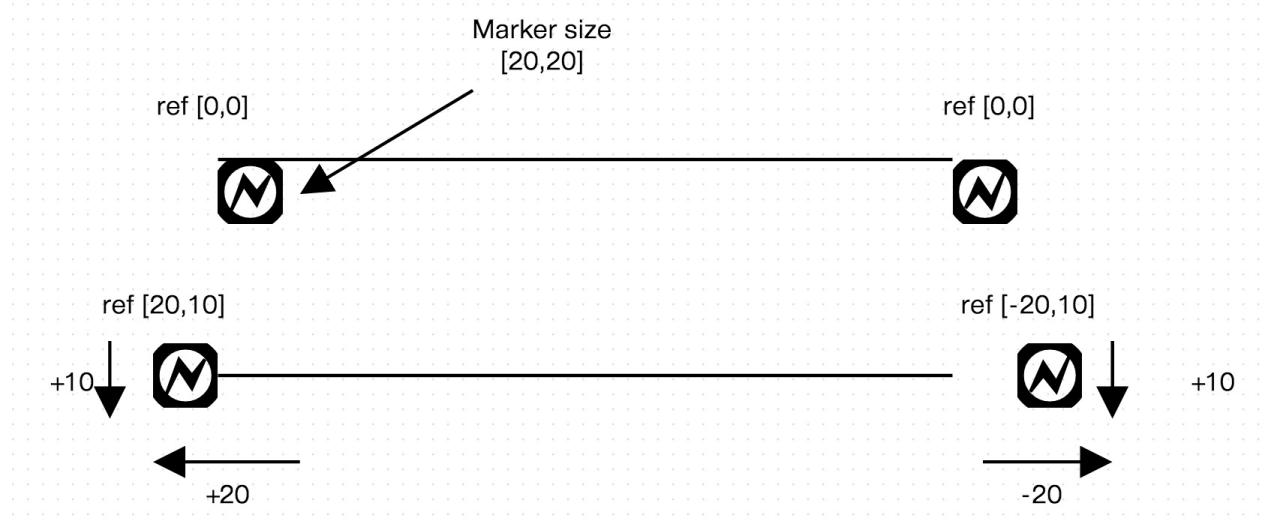
계속해서, 마커에 ref 를 적용해보도록 하겠습니다.

ref 는 선 경로상의 마커 위치를 뜻하는 것으로, ref 를 지정하지 않으면 오픈그래프가 마커의 사이즈에 맞추어서 자동으로 ref 를 배정해줍니다.

ref 를 직접 적용하여 아래 코드를 작성하도록 하고, ref 의 위치 지정 방식을 그림으로 살펴보고 이해해도록 합니다.

ref 의 x 좌표는 경로상을 기준으로 삼다보니, 실제 좌표계에서의 x 충분 값과 반대가 되니 혼동되지 않도록 합니다.

```
var edge1 = canvas.drawShape(null, new OG.EdgeShape([100, 100], [500, 100]),  
null);  
var edge2 = canvas.drawShape(null, new OG.EdgeShape([100, 200], [500, 200]),  
null);  
  
canvas.setShapeStyle(edge1, {  
    'marker': {  
        'start': {  
            'id': 'OG.marker.RectangleMarker',  
            'size': [20, 20],  
            'ref': [0,0]  
        },  
        'end': {  
            'id': 'OG.marker.RectangleMarker',  
            'size': [20, 20],  
            'ref': [0,0]  
        }  
    }  
});  
  
canvas.setShapeStyle(edge2, {  
    'marker': {  
        'start': {  
            'id': 'OG.marker.RectangleMarker',  
            'size': [20, 20],  
            'ref': [20,10]  
        },  
        'end': {  
            'id': 'OG.marker.RectangleMarker',  
            'size': [20, 20],  
            'ref': [0,10]  
        }  
    }  
});
```

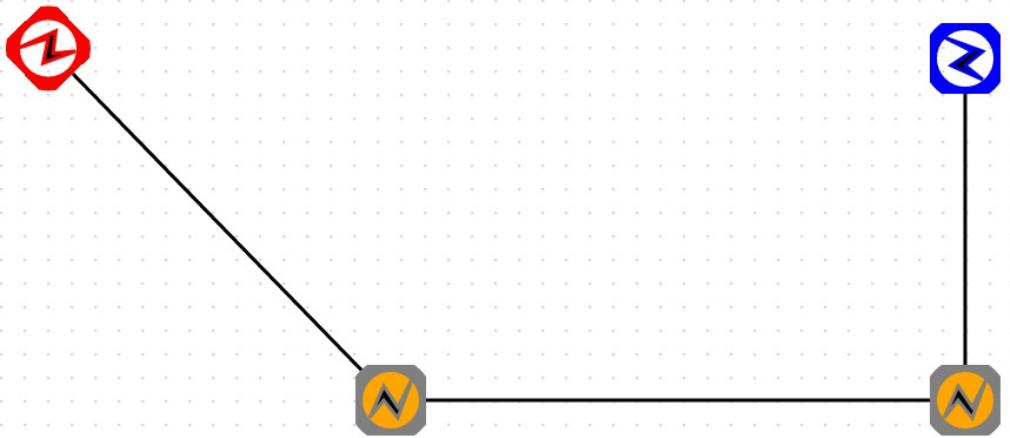


## Custom Marker Style

제작한 커스텀 마커를 적용시킬 때, 특정 마커마다의 스타일을 다르게 적용시킬 수 있습니다.

```
var edge1 = canvas.drawShape(null, new OG.EdgeShape([100, 100], [500, 100]),
null);

canvas.setShapeStyle(edge1, {
  'marker': {
    'start': {
      'id': 'OG.marker.RectangleMarker',
      'size': [20, 20],
      'style':{
        'stroke': 'red'
      }
    },
    'mid': {
      'id': 'OG.marker.RectangleMarker',
      'size': [20, 20],
      'style':{
        'stroke': 'gray',
        'fill': 'orange',
        'fill-opacity': 'orange'
      }
    },
    'end': {
      'id': 'OG.marker.RectangleMarker',
      'size': [20, 20],
      'style':{
        'stroke': 'blue'
      }
    }
  }
});
```



# Pattern

- [Custom Pattern](#)
- [Custom Pattern Size](#)
- [Custom Pattern Unit Size](#)
- [Custom Pattern Style && Transform](#)
- [Hatched Pattern](#)
- [Custom Pattern in EdgeShape](#)

패턴 스타일은 도형의 내부의 fill 어트리뷰트를 대체하며, 도형 속에 특정 패턴을 반복하여 드로잉합니다.

## [SVG Fill Patterns](#)

## Custom Pattern

오픈그래프는 사용자 스스로 패턴을 제작하여 스타일링에 활용할 수 있도록 지원하고, 기본 제공 패턴은 지원하지 않습니다.

커스텀 패턴을 사용하기 위해서는, 우선 패턴 클래스 하나를 제작해보도록 합니다.

패턴 클래스 제작은 [Define Custom Shape \(extend-shape.md#define-custom-shape\)](#) 제작방식과 동일하지만 네임스페이스만 바뀝니다.

다음은 사각형 모양의 패턴 클래스 샘플입니다.

```

/**
 * Rect Pattern
 *
 * @class
 * @extends OG.pattern.IPattern
 * @requires OG.common.*
 * @requires OG.geometry.*
 *
 * @author <a href="mailto:sppark@uengine.org">Seungpil Park</a>
 */
OG.pattern.RectPattern = function () {
    OG.pattern.RectPattern.superclass.call(this);

    this.PATTERN_ID = 'OG.pattern.RectPattern';
};

OG.pattern.RectPattern.prototype = new OG.pattern.IPattern();
OG.pattern.RectPattern.superclass = OG.pattern.IPattern;
OG.pattern.RectPattern.prototype.constructor = OG.pattern.RectPattern;
OG.RectPattern = OG.pattern.RectPattern;

/**
 * 드로잉할 pattern 을 생성하여 반환한다.
 *
 * @return {OG.geometry.Geometry} pattern 정보
 * @override
 */
OG.pattern.RectPattern.prototype.createPattern = function () {
    if (this.geom) {
        return this.geom;
    }

    this.geom = new OG.geometry.Rectangle([0, 0], 100, 100);
    return this.geom;
};

```

## Custom Pattern Size

커스텀 Shape 를 생성할때와 모든 코드가 동일하지만, OG.shape 가 OG.pattern 으로 바뀌었음을 알 수 있습니다.

이제 이 코드를 오픈그래프 라이브러리를 불러온 후 html 페이지에 삽입하도록 합니다.

다음 할 일은 이 패턴을 도형에 적용시키는 것입니다.

도형에 적용시킬 때는 "pattern" 스타일 어트리뷰트를 사용하게 되는데, 몇가지 프로퍼티와 함께 json 형식으로 표현하도록 합니다.

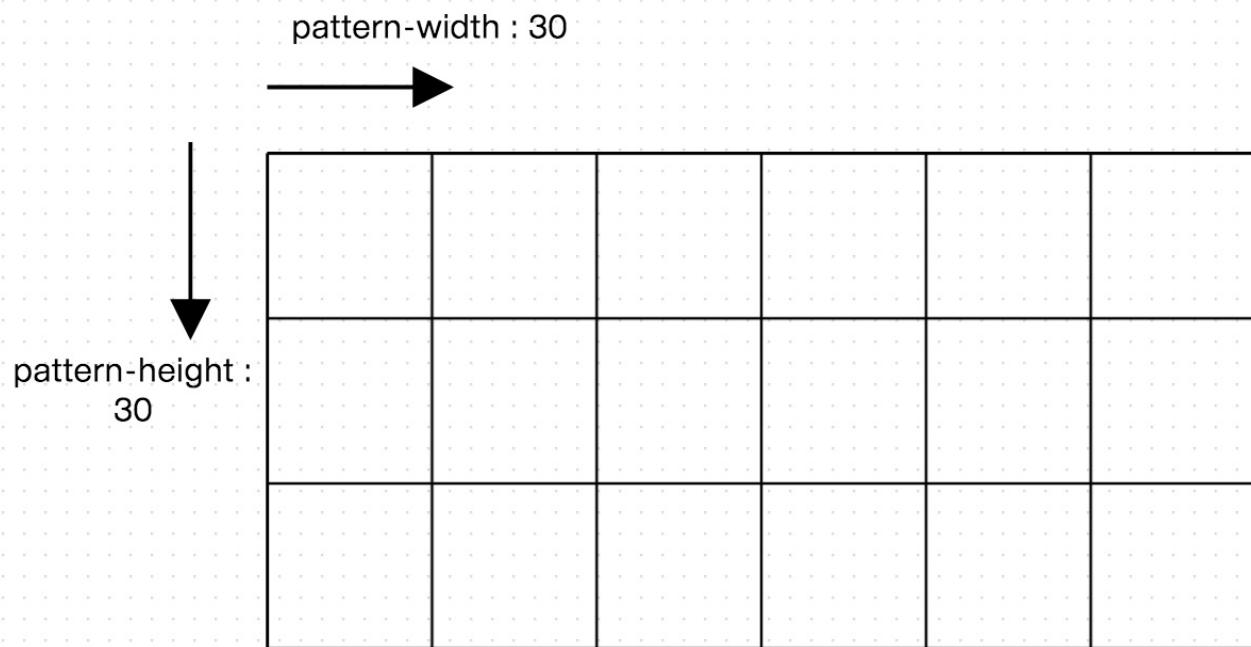
패턴 프로퍼티	설명	설명	비고
id	string	패턴 클래스명	
pattern-width	number	패턴 가로	
pattern-height	number	패턴 세로	
patternTransform	number	패턴의 transform	Svg transform syntax

unit-width	number 패턴 유닛의 가로
unit-height	number 패턴 유닛의 세로
thickness	number 선 도형에 패턴을 채울경우 패턴 두께 EdgeShape 도형 한정
style	Object 패턴 유닛 스타일

위의 표를 참조하여 사각형의 도형 안에 OG.pattern.RectPattern 패턴을 적용하도록 해봅니다.

```
var rect = canvas.drawShape([200,200], new OG.RectangleShape(), [300,200]);

canvas.setShapeStyle(rect, {
    pattern: {
        'id': 'OG.pattern.RectPattern',
        'thickness': 10,
        'pattern-width': 60,
        'pattern-height': 60,
    },
    'fill-opacity':1
});
```



## Custom Pattern Unit Size

계속해서, 마커에 unit-width 와 unit-height 를 적용해보도록 하겠습니다.

pattern-width 와 pattern-size 는 도형안에의 하나의 패턴이 차지하는 블록 영역이라면, unit 은 블록안에서 차지하는 패턴의 geometry 영역을 뜻합니다.

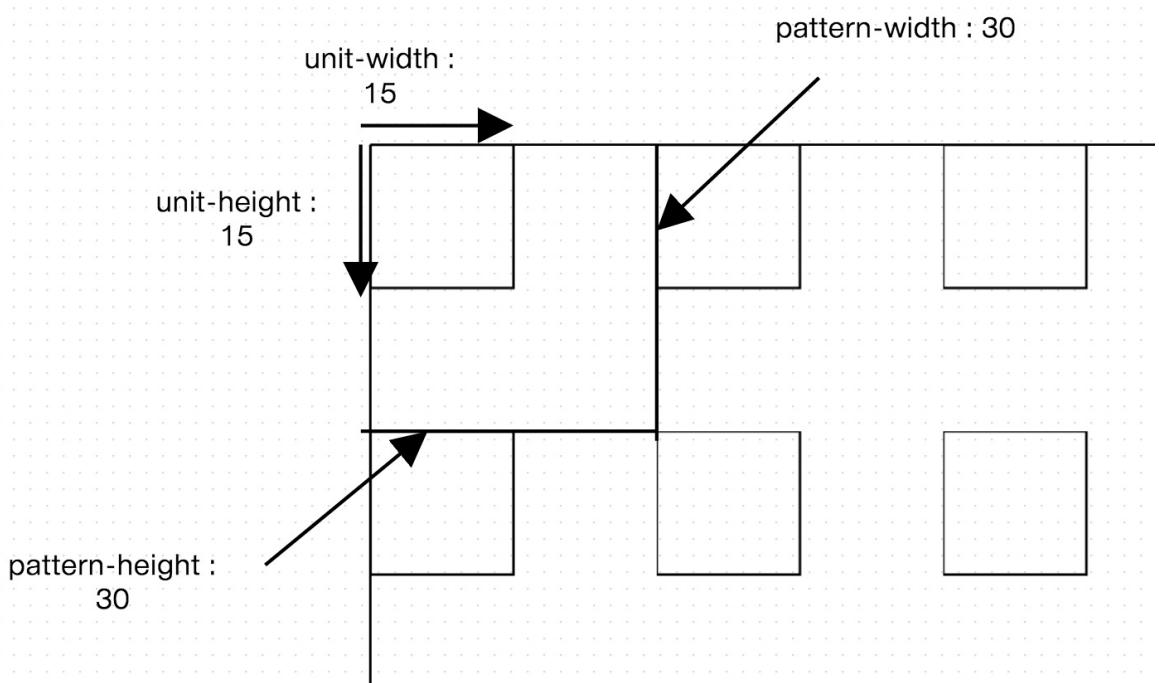
unit-width 와 unit-height 를 직접 적용하여 아래 코드를 실행시키고 이해하여 보도록 합니다.

```

var rect = canvas.drawShape([200,200], new OG.RectangleShape(), [300,200]);

canvas.setShapeStyle(rect, {
    pattern: {
        'id': 'OG.pattern.RectPattern',
        'unit-width': 15,
        'unit-height': 15,
        'pattern-width': 30,
        'pattern-height': 30
    },
    'fill-opacity':1
});

```



## Custom Pattern Style && Transform

제작한 커스텀 패턴을 적용시킬 때, `style` 프로퍼티로 패턴에 스타일을 적용할 수 있습니다.

`patternTransform` 은 패턴 각각의 `transform` 을 의미하는것 아닌, 전체적 패턴 배열의 `transform` 을 정의합니다.

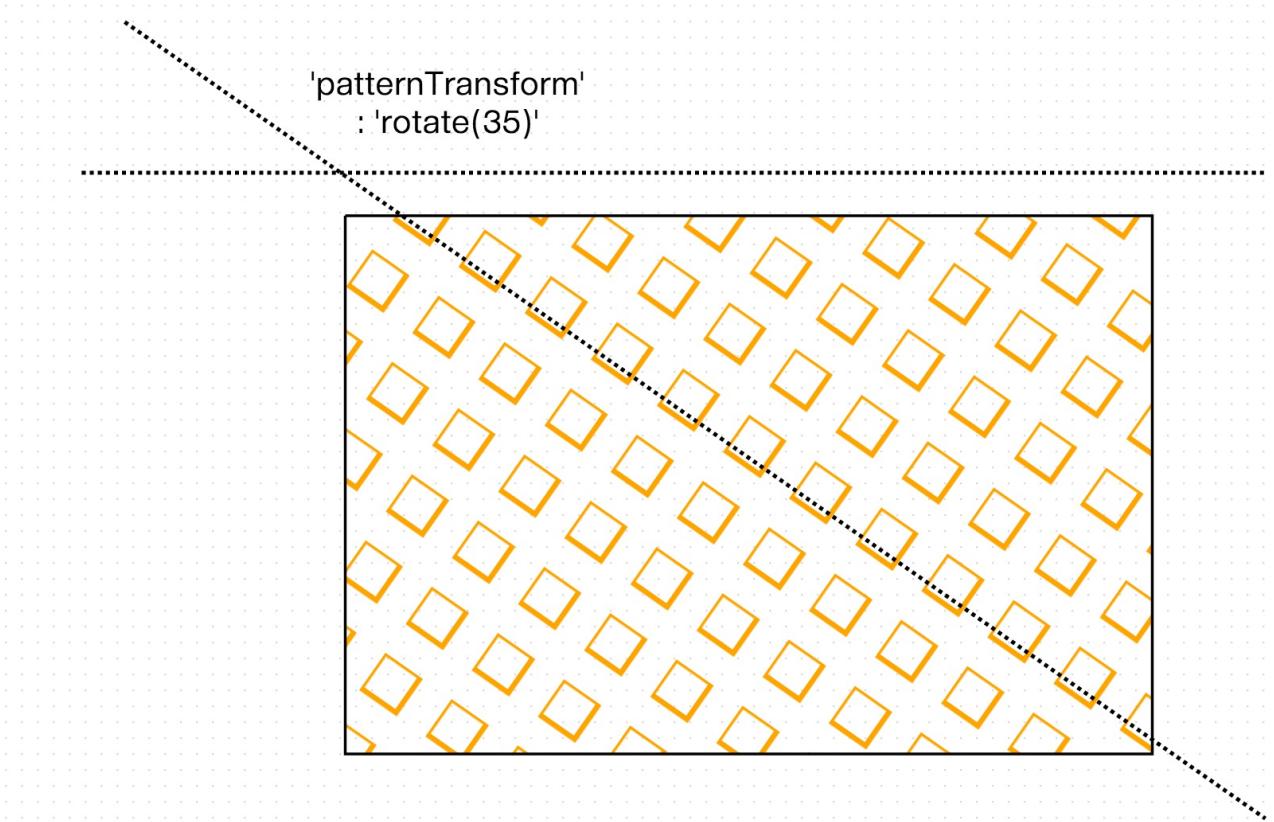
아래 예제를 통해 `patternTransform` 과 `style` 을 적용해보도록 합니다.

```

var rect = canvas.drawShape([200,200], new OG.RectangleShape(), [300,200]);

canvas.setShapeStyle(rect, {
    pattern: {
        'id': 'OG.pattern.RectPattern',
        'pattern-width': 30,
        'pattern-height': 30,
        'unit-width': 15,
        'unit-height': 15,
        'patternTransform': 'rotate(35)',
        'style': {
            'stroke-width': 2,
            'stroke': 'orange'
        }
    },
    'fill-opacity':1
});

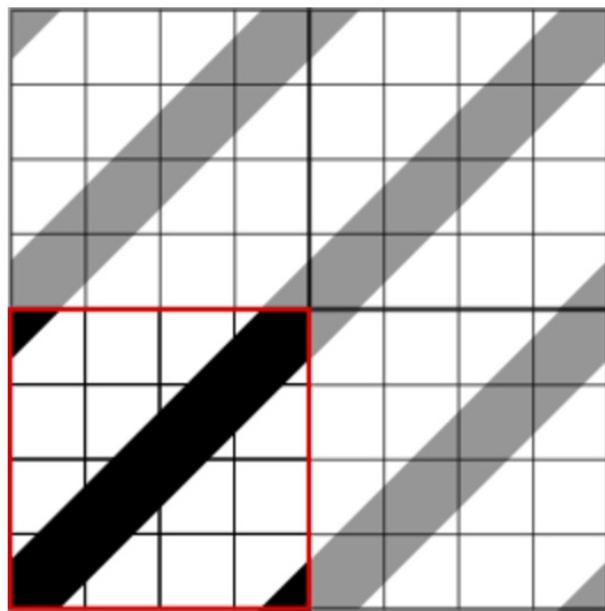
```



## Hatched Pattern

패턴 클래스 제작시 독립적인 모양의 패턴이 아닌 패턴끼리 연결되어 특정 모양을 나타내야 할 경우가 있습니다.

아래 그림과 같은 Hatched (빗금) 모양 패턴이 좋은 예입니다.



위의 그림에서처럼, 연속된 모양의 빗금 처리를 위해서는 하나의 패턴에 세 줄의 라인이 있어야 합니다.

위의 모양을 구현하기 위한 `geometry` 처리를 하는 패턴 클래스를 제작해보도록 합니다.

```

/**
 * Hatched Pattern
 *
 * @class
 * @extends OG.pattern.IPattern
 * @requires OG.common.*
 * @requires OG.geometry.*
 *
 * @author <a href="mailto:sppark@uengine.org">Seungpil Park</a>
 */
OG.pattern.HatchedPattern = function () {
    OG.pattern.HatchedPattern.superclass.call(this);

    this.PATTERN_ID = 'OG.pattern.HatchedPattern';
};

OG.pattern.HatchedPattern.prototype = new OG.pattern.IPattern();
OG.pattern.HatchedPattern.superclass = OG.pattern.IPattern;
OG.pattern.HatchedPattern.prototype.constructor = OG.pattern.HatchedPattern;
OG.HatchedPattern = OG.pattern.HatchedPattern;

/**
 * 드로잉할 pattern 을 생성하여 반환한다.
 *
 * @return {OG.geometry.Geometry} pattern 정보
 * @override
 */
OG.pattern.HatchedPattern.prototype.createPattern = function () {
    var geom1, geom2, geom3, geomCollection = [];
    if (this.geom) {
        return this.geom;
    }

    geom1 = new OG.geometry.Line([-1, 1], [1, -1]);
    geom2 = new OG.geometry.Line([0, 4], [4, 0]);
    geom3 = new OG.geometry.Line([3, 5], [5, 3]);

    geomCollection.push(geom1);
    geomCollection.push(geom2);
    geomCollection.push(geom3);

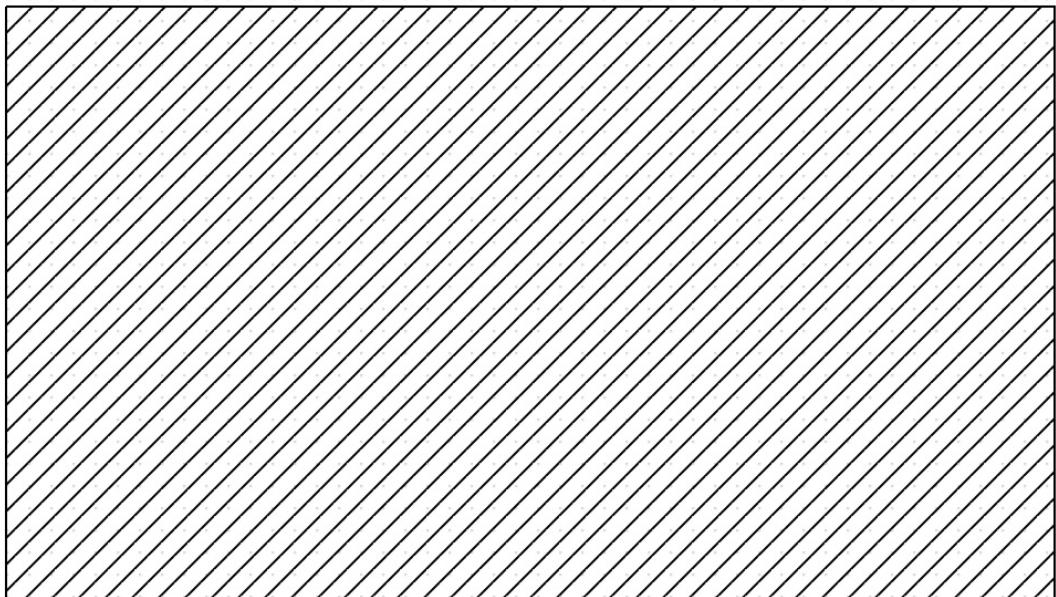
    this.geom = new OG.geometry.GeometryCollection(geomCollection);

    return this.geom;
};

```

위 패턴 클래스를 이용해 도형을 그릴 경우 빗금 모양의 패턴으로 채워지게 됩니다.

```
var rect = canvas.drawShape([200, 200], new OG.RectangleShape(), [300, 200]);  
  
canvas.setShapeStyle(rect, {  
    pattern: {  
        'id': 'OG.pattern.HatchedPattern',  
        'unit-width': 18,  
        'unit-height': 18,  
        'pattern-width': 12,  
        'pattern-height': 12,  
        'style': {  
            'stroke': 'black'  
        }  
    },  
    'fill-opacity': 1  
});
```



## Custom Pattern in EdgeShape

오픈그래프에서는 선 도형에도 패턴을 적용시킬 수 있도록 지원합니다.

패턴 스타일을 EdgeShape 선도형에 주게 될 경우, 패턴 두께 프로퍼티인 thickness 를 추가로 지정해야 합니다.

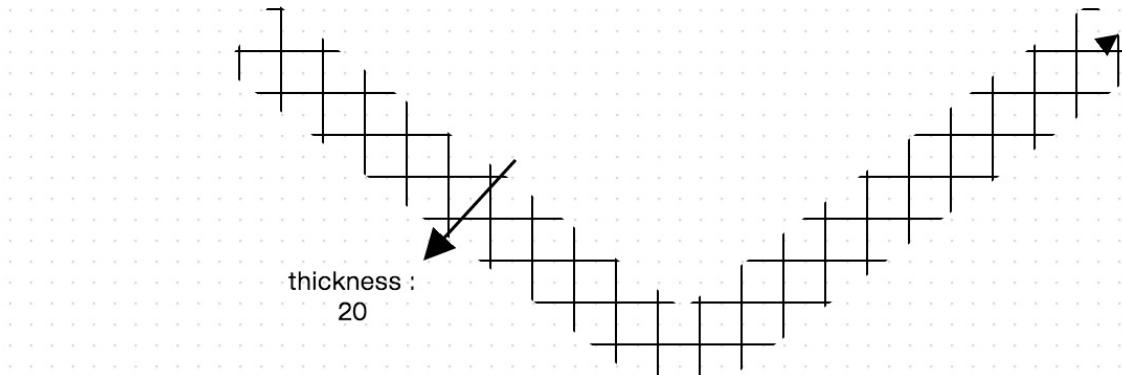
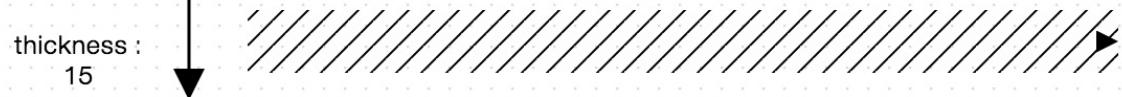
thickness 를 추가하여 샘플 코드를 작성해보도록 합니다.

```

var edge1 = canvas.drawShape(null, new OG.EdgeShape([100, 100], [500, 100]),
null);
var edge2 = canvas.drawShape(null, new OG.EdgeShape([100, 150], [500, 150]),
null);

canvas.setShapeStyle(edge1, {
    pattern: {
        'id': 'OG.pattern.HatchedPattern',
        'thickness': 15,
        'unit-width': 18,
        'unit-height': 18,
        'pattern-width': 12,
        'pattern-height': 12,
        'style': {
            'stroke': 'black'
        }
    },
    'stroke': 'none'
});
canvas.setShapeStyle(edge2, {
    pattern: {
        'id': 'OG.pattern.RectPattern',
        'thickness': 20,
        'pattern-width': 20,
        'pattern-height': 20,
        'style': {
            'stroke': 'black'
        }
    },
    'stroke': 'none'
});

```





# Multi Line Edge

- [Basic Multi Edge](#)
- [Multi Edge from to](#)
- [Multi Edge Examples](#)

## Basic Multi Edge

오픈그래프의 EdgeShape 를 드로잉할 때, 기본적으로 하나의 선분으로 표현됩니다.

도형 "multi" 스타일 어트리뷰트를 통해 다수의 선분으로 이루어진 연결선 표현이 가능한데, 각각의 선분마다 개별적인 스타일 및 마커를 적용할 수 있습니다.

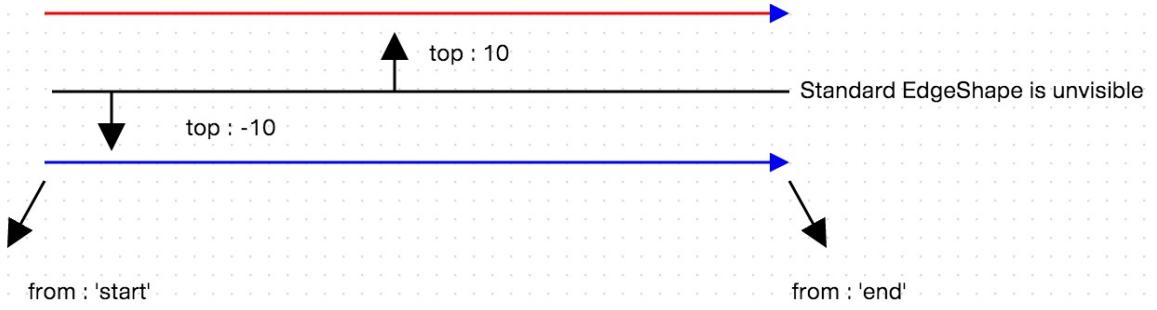
또한, "multi" 스타일 어트리뷰트가 붙은 선분은 스스로는 보이지 않게 되고, multi 안에 표현된 선분만 드로잉됩니다.

"multi" 스타일 어트리뷰트는 몇가지 프로퍼티와 함께 json 형식으로 표현하도록 합니다.

프로퍼티	설명	설명	비고
top	number	기준선으로부터의 거리	
from	number or string	기준선 대비 선분의 시작점 10,'10','10px','10%','start','center','end','end-10'	
to	number or string	기준선 대비 선분의 종료점 10,'10','10px','10%','start','center','end','end-10'	
style	Object	선분의 스타일	

위의 표를 참조하여 두 줄로 이루어진 EdgeShape 를 생성해보도록 합니다.

```
var edge1 = canvas.drawShape(null, new OG.EdgeShape([100, 100], [500, 100]),  
null);  
  
canvas.setShapeStyle(edge1, {  
    multi: [  
        {  
            'top': 10,  
            'from': 'start',  
            'to': 'end',  
            'style': {  
                'arrow-end': 'block',  
                'stroke': 'red'  
            }  
        },  
        {  
            'top': -10,  
            'from': 'start',  
            'to': 'end',  
            'style': {  
                'arrow-end': 'block',  
                'stroke': 'blue'  
            }  
        }  
    ]  
});
```



## Multi Edge from to

from 과 to 프로퍼티는 선분이 가진 Path로부터 시작점, 종료점을 의미합니다.

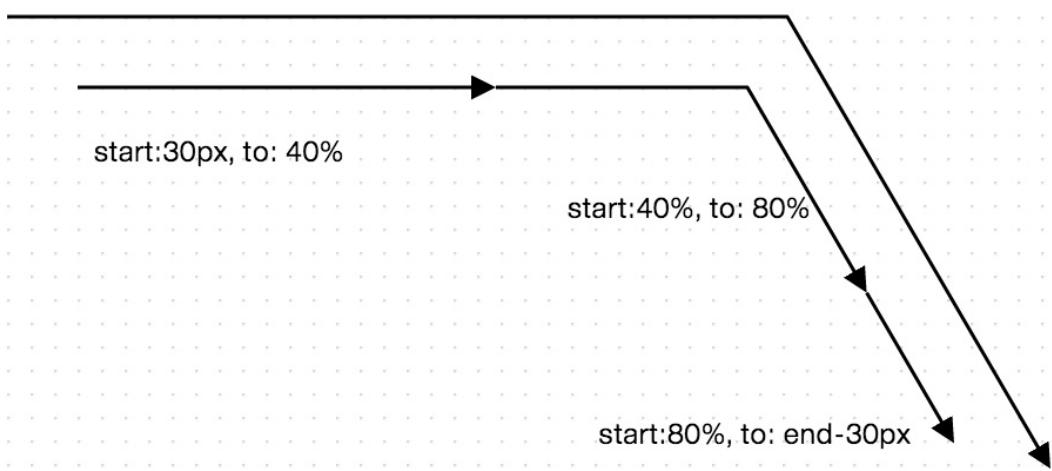
표현방식은 px, number, percentage, end-number 방식으로 가능합니다.

표현식	설명
10	시작점으로부터 10px
10	시작점으로부터 10px
10px	시작점으로부터 10px
10%	시작점으로부터 10퍼센테이지
start	시작점
center	중앙점
end	끝점
end-10	끝점으로부터 10px

위의 표현식으로 아래의 샘플 코드를 구현해보도록 합니다.

```
var edge1 = canvas.drawShape(null, new OG.EdgeShape([100, 100], [500, 100]), null);

canvas.setShapeStyle(edge1, {
    multi: [
        {
            'top': 0,
            'from': 'start',
            'to': 'end',
            'style': {
                'arrow-end': 'block'
            }
        },
        {
            'top': -30,
            'from': '30px',
            'to': '40%',
            'style': {
                'arrow-end': 'block'
            }
        },
        {
            'top': -30,
            'from': '40%',
            'to': '80%',
            'style': {
                'arrow-end': 'block'
            }
        },
        {
            'top': -30,
            'from': '80%',
            'to': 'end-30px',
            'style': {
                'arrow-end': 'block'
            }
        }
    ]
});
```



## Multi Edge Examples

EdgeShape 를 상속받는 선 도형 클래스를 만들고, 도형간의 연결에 활용해보도록 하겠습니다.

```
/**
 * ELECTRONIC : CableShape
 *
 * @class
 * @extends OG.shape.CableShape
 * @requires OG.common.*
 * @requires OG.geometry.*
 *
 * @param {Number[]} from 와이어 시작 좌표
 * @param {Number[]} to 와이어 끝 좌표
 * @param {String} label 라벨 [Optional]
 * @author <a href="mailto:sppark@uengine.org">Seungpil Park</a>
 * @private
 */
OG.shape.elec.CableShape = function (from, to, label) {
    OG.shape.elec.CableShape.superclass.call(this, from, to, label);

    this.SHAPE_ID = 'OG.shape.elec.CableShape';
};

OG.shape.elec.CableShape.prototype = new OG.shape.EdgeShape();
OG.shape.elec.CableShape.superclass = OG.shape.EdgeShape;
OG.shape.elec.CableShape.prototype.constructor = OG.shape.elec.CableShape;
OG.CableShape = OG.shape.elec.CableShape;

/**
 * 드로잉할 Shape 을 생성하여 반환한다.
 *
 * @return {OG.geometry.Geometry} Shape 정보
 * @override
 */
```

```

OG.shape.elec.CableShape.prototype.createShape = function () {
    if (this.geom) {
        return this.geom;
    }

    this.geom = new OG.Line(this.from || [0, 0], this.to || [70, 0]);
    this.geom.style = new OG.geometry.Style({
        'multi': [
            {
                top: 0,
                from: 'start',
                to: 'center',
                style: {
                    'marker': {
                        'end': {
                            'id': 'OG.marker.SwitchLMarker',
                            'size': [20, 8],
                            'ref': [3, 0]
                        }
                    }
                }
            },
            {
                top: 0,
                from: 'center',
                to: 'end',
                style: {
                    'marker': {
                        'start': {
                            'id': 'OG.marker.SwitchXMarker',
                            'size': [6, 6]
                        }
                    }
                }
            }
        ]
    });
    return this.geom;
};

/**
 * ELECTRONIC : Busduct Shape
 *
 * @class
 * @extends OG.shape.BusductShape
 * @requires OG.common.*
 * @requires OG.geometry.*
 *
 * @param {Number[]} from 와이어 시작 좌표
 * @param {Number[]} to 와이어 끝 좌표
 * @param {String} label 라벨 [Optional]
 */

```

```

* @author <a href="mailto:sppark@uengine.org">Seungpil Park</a>
* @private
*/
OG.shape.elec.BusductShape = function (from, to, label) {
    OG.shape.elec.BusductShape.superclass.call(this, from, to, label);

    this.SHAPE_ID = 'OG.shape.elec.BusductShape';
};

OG.shape.elec.BusductShape.prototype = new OG.shape.EdgeShape();
OG.shape.elec.BusductShape.superclass = OG.shape.EdgeShape;
OG.shape.elec.BusductShape.prototype.constructor = OG.shape.elec.BusductShape;
OG.BusductShape = OG.shape.elec.BusductShape;

/**
 * 드로잉할 Shape 을 생성하여 반환한다.
 *
 * @return {OG.geometry.Geometry} Shape 정보
 * @override
*/
OG.shape.elec.BusductShape.prototype.createShape = function () {
    if (this.geom) {
        return this.geom;
    }

    this.geom = new OG.Line(this.from || [0, 0], this.to || [70, 0]);
    this.geom.style = new OG.geometry.Style({
        'multi': [
            {
                top: -10,
                from: '20px',
                to: 'end-20px',
                style: {
                    'marker': {
                        'start': {
                            'id': 'OG.marker.SwitchRMarker',
                            'size': [8, 8],
                            'ref': [8, 0]
                        },
                        'end': {
                            'id': 'OG.marker.SwitchLMarker',
                            'size': [8, 8],
                            'ref': [1, 0]
                        }
                    }
                }
            },
            {
                top: 10,
                from: '20px',
                to: 'end-20px',
                style: {

```

```

        'marker': {
            'start': {
                'id': 'OG.marker.SwitchLMarker',
                'size': [8, 8],
                'ref': [7, 8]
            },
            'end': {
                'id': 'OG.marker.SwitchRMarker',
                'size': [8, 8],
                'ref': [0, 8]
            }
        }
    },
    {
        top: 0,
        from: 'start',
        to: 'end',
        style: {}
    }
]
);
return this.geom;
};

/**
* ELECTRONIC : Raceway Shape
*
* @class
* @extends OG.shape.RacewayShape
* @requires OG.common.*
* @requires OG.geometry.*
*
* @param {Number[]} from 와이어 시작 좌표
* @param {Number[]} to 와이어 끝 좌표
* @param {String} label 라벨 [Optional]
* @author <a href="mailto:sppark@uengine.org">Seungpil Park</a>
* @private
*/
OG.shape.elec.RacewayShape = function (from, to, label) {
    OG.shape.elec.RacewayShape.superclass.call(this, from, to, label);

    this.SHAPE_ID = 'OG.shape.elec.RacewayShape';
};

OG.shape.elec.RacewayShape.prototype = new OG.shape.EdgeShape();
OG.shape.elec.RacewayShape.superclass = OG.shape.EdgeShape;
OG.shape.elec.RacewayShape.prototype.constructor = OG.shape.elec.RacewayShape;
OG.RacewayShape = OG.shape.elec.RacewayShape;

/**
* 드로잉할 Shape 을 생성하여 반환한다.

```

```

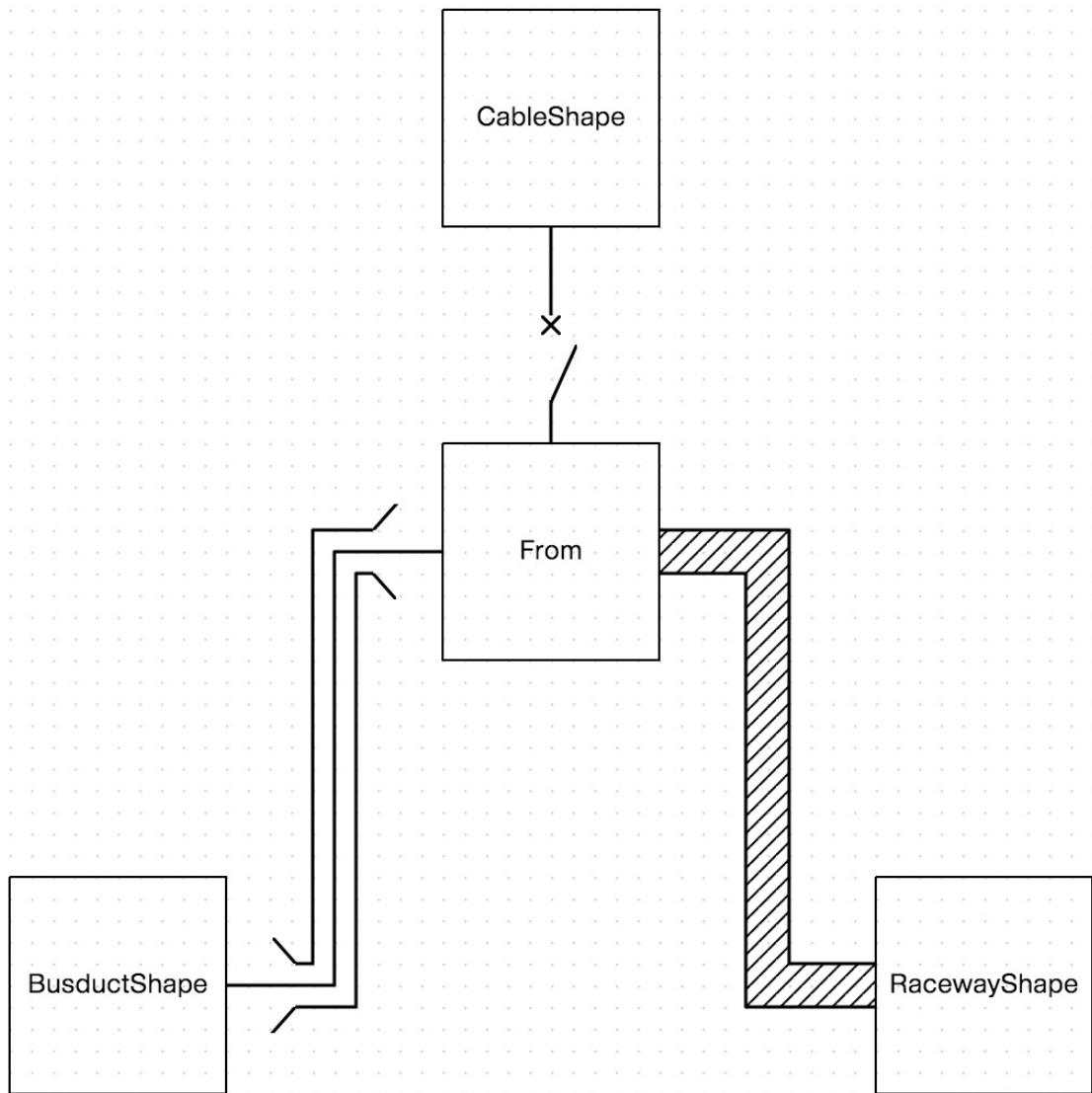
/*
 * @return {OG.geometry.Geometry} Shape 정보
 * @override
 */
OG.shape.elec.RacewayShape.prototype.createShape = function () {
    if (this.geom) {
        return this.geom;
    }

    this.geom = new OG.Line(this.from || [0, 0], this.to || [70, 0]);
    this.geom.style = new OG.geometry.Style({
        'multi': [
            {
                top: -10,
                from: 'start',
                to: 'end',
                style: {}
            },
            {
                top: 10,
                from: 'start',
                to: 'end',
                style: {}
            },
            {
                top: 0,
                from: 'start',
                to: 'end',
                style: {
                    pattern: {
                        'id': 'OG.pattern.HatchedPattern',
                        'thickness': 10,
                        'unit-width': 12,
                        'unit-height': 12,
                        'pattern-width': 8,
                        'pattern-height': 8,
                        'style': {
                            'stroke': 'black'
                        }
                    },
                    'stroke': 'none'
                }
            }
        ]
    });
    return this.geom;
};

var rect1 = canvas.drawShape([400, 400], new OG.RectangleShape('From'), [100, 100]);
var rect2 = canvas.drawShape([400, 200], new OG.RectangleShape('CableShape'),
```

```
[100, 100]);
var rect3 = canvas.drawShape([200, 600], new OG.RectangleShape('BusductShape'),
[100, 100]);
var rect4 = canvas.drawShape([600, 600], new OG.RectangleShape('RacewayShape'),
[100, 100]);

canvas.connect(rect1, rect2, null, null, null, null, null, null, new
OG.CableShape());
canvas.connect(rect1, rect3, null, null, null, null, null, null, new
OG.BusductShape());
canvas.connect(rect1, rect4, null, null, null, null, null, null, new
OG.RacewayShape());
```



# Animation

애니메이션은 스타일 어트리뷰트의 변화를 시나리오로 작성하여 도형이 그려지는 시점에 애니메이팅이 되도록 해줍니다.

애니메이션은 스타일의 "animation" 어트리뷰트와 "animation-repeat" 어트리뷰트를 사용합니다.

어트리뷰트	프로퍼티	타입	설명
animation	start	Object	애니메이션 시작 스타일
	to	Object	애니메이션 종료 스타일
	ms	number	플레이 시간
	delay	number	플레이까지 대기시간
animation-repeat		boolean	반복 여부

등록된 애니메이션 시나리오들은 모두 동시 시작하므로, 순차 시작을 원할 경우 delay 값으로 조절해야 합니다.

또한 animation-repeat 어트리뷰트를 통해 반복 재생 여부를 결정할 수 있습니다.

아래 예제를 진행한다면, 직사각형 도형의 색상이 처음 1초간은 white에서 black으로, 그다음 1초간은 black에서 white로 반복재생 될 것 입니다.

```
var rect1 = canvas.drawShape([200, 200], new OG.RectangleShape(), [100, 100],  
    {  
        animation: [  
            {  
                start: {  
                    fill: 'white'  
                },  
                to: {  
                    fill: 'black'  
                },  
                ms: 1000  
            },  
            {  
                start: {  
                    fill: 'black'  
                },  
                to: {  
                    fill: 'white'  
                },  
                ms: 1000,  
                delay: 1000  
            }  
        ],  
        'animation-repeat':1,  
        'fill-opacity': 1  
    });
```

# Sub Shape

- [Basic Sub Shape](#)
- [Sub Shape Size && Position](#)
- [Sub Shape index](#)
- [Sub Shape Data model](#)

앞서 [Define Custom Shape \(extend-shape.md#define-custom-shape\)](#) 파트를 통해 복잡한 기하학 도형을 표현하는 것을 살펴보았습니다.

Sub Shape 기능이 위와 차이가 있다면 위와 같은 도형은 정해진 geometry 대로 도형을 그리지만, Sub Shape 기능은 다수의 Shape 을 도형의 상태 또는 데이터에 따라 유기적으로 배치할 수 있도록 모델링 한 도형입니다.

## Basic Sub Shape

Sub Shape 은 도형 클래스 정의시 createSubShape 메소드를 정의함으로써 사용할 수 있습니다.

먼저, 베이스가 될 라운드 사각형 모양의 도형 클래스를 정의해보도록 합니다.

```

OG.shape.SampleShape = function (label) {
    OG.shape.SampleShape.superclass.call(this);

    this.GROUP_DROPABLE = false;
    this.SHAPE_ID = 'OG.shape.SampleShape';
    this.label = label;
    this.CONNECTABLE = true;
    this.GROUP_COLLAPSIBLE = false;
    this.LoopType = "None";
    this.TaskType = "None";
    this.status = "None";
    this.Events = [];

};

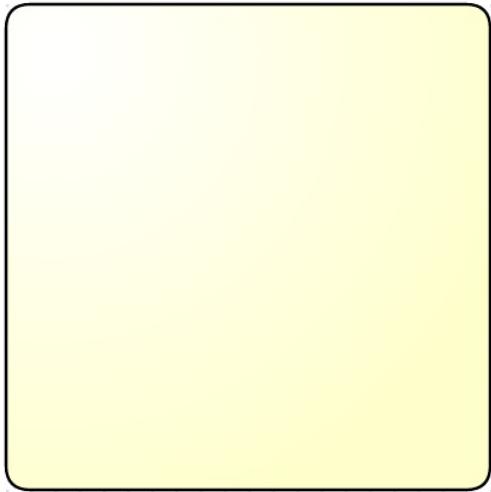
OG.shape.SampleShape.prototype = new OG.shape.GemShape();
OG.shape.SampleShape.superclass = OG.shape.GemShape;
OG.shape.SampleShape.prototype.constructor = OG.shape.SampleShape;
OG.SampleShape = OG.shape.SampleShape;

/**
 * 드로잉할 Shape 을 생성하여 반환한다.
 *
 * @return {OG.geometry.Geometry} Shape 정보
 * @override
 */
OG.shape.SampleShape.prototype.createShape = function () {
    if (this.geom) {
        return this.geom;
    }

    this.geom = new OG.geometry.Rectangle([0, 0], 100, 100);
    this.geom.style = new OG.geometry.Style({
        'fill-r': 1,
        'fill-cx': .1,
        'fill-cy': .1,
        "stroke-width": 1.2,
        fill: 'r(.1, .1)FFFFF-#FFFFCC',
        'fill-opacity': 1,
        r: '10'
    });
    return this.geom;
};

canvas.drawShape([300,200],new OG.SampleShape(), [200,200]);

```



이제 이 클래스에 `createSubShape` 메소드를 정의합니다.

`createSubShape` 메소드는 `this.sub` 객체를 리턴하도록 하는데, `this.sub` 안에는 추가적으로 그려야 할 sub shapes 들을 기술합니다.

기술 내용에 대한 프로퍼티는 다음과 같습니다.

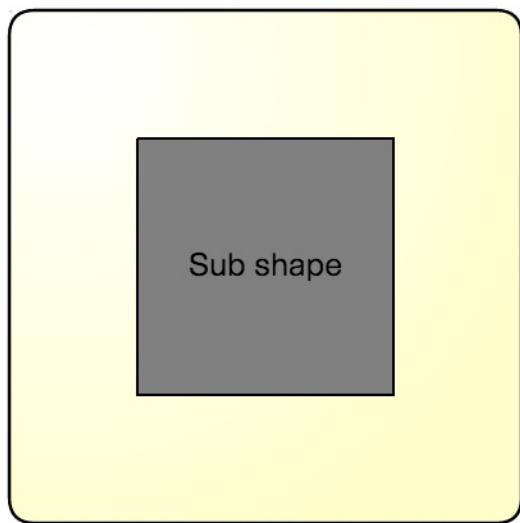
프로퍼티	타입	설명
<code>shape</code>	<code>OG.shape.IShape</code>	서브 도형 클래스
<code>width</code>	<code>number or string</code>	서브 도형 가로
<code>height</code>	<code>number or string</code>	서브 도형 세로
<code>top</code>	<code>number or string</code>	서브 도형 top 포지션
<code>bottom</code>	<code>number or string</code>	서브 도형 bottom 포지션
<code>left</code>	<code>number or string</code>	서브 도형 left 포지션
<code>right</code>	<code>number or string</code>	서브 도형 right 포지션
<code>align</code>	<code>string</code>	가로 정렬 (left,center,right)
<code>vertical-align</code>	<code>string</code>	세로 정렬 (top,middle,bottom)
<code>style</code>	<code>Object</code>	서브 도형 스타일

```

OG.shape.SampleShape.prototype.createSubShape = function () {
    this.sub = [
        {
            shape: new OG.RectangleShape('Sub shape'),
            width: '50%',
            height: '50%',
            align: 'center',
            'vertical-align': 'middle',
            style: {
                fill: 'gray',
                'fill-opacity': 1
            }
        }
    ];
    return this.sub;
};

canvas.drawShape([300,200],new OG.SampleShape(), [200,200]);

```



## Sub Shape Size && Position

서브 도형의 사이즈와 포지션은 CSS 어트리뷰트 표기법과 동일하게 표현할 수 있습니다.

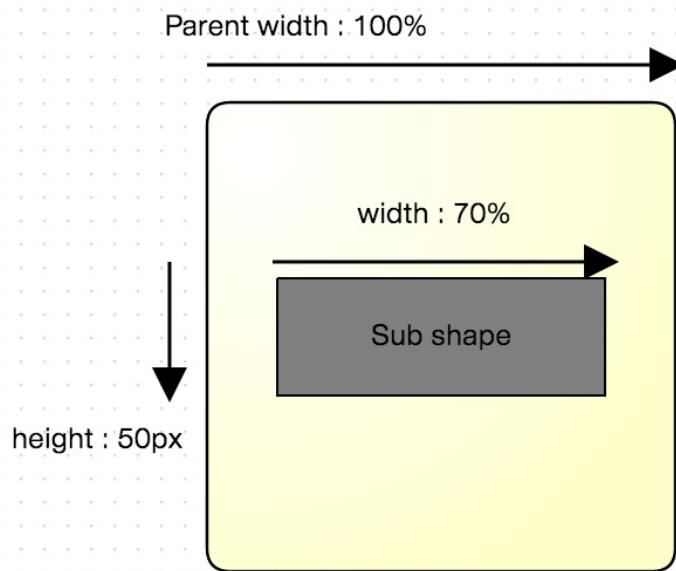
서브 도형의 사이즈는 `width` 와 `height` 프로퍼티로 결정합니다. `px` 로 표기할 경우 표기한 `px` 만큼 표현되고, `%` 로 표기할 경우 부모 도형의 가로, 세로 대비 `%` 로 계산되어 표현됩니다.

```

OG.shape.SampleShape.prototype.createSubShape = function () {
    this.sub = [
        {
            shape: new OG.RectangleShape('Sub shape'),
            width: '70%',
            height: '50px',
            align: 'center',
            'vertical-align': 'middle',
            style: {
                fill: 'gray',
                'fill-opacity': 1
            }
        }
    ];
    return this.sub;
};

canvas.drawShape([300,200],new OG.SampleShape(), [200,200]);

```



서브 도형의 위치는 CSS position 어트리뷰트 중, "absolute" 위치 계산과 동일합니다.

```

OG.shape.SampleShape.prototype.createSubShape = function () {
    this.sub = [
        {
            shape: new OG.RectangleShape('Sub1'),
            width: '50px',
            height: '50px',
            align: 'left',
            'vertical-align': 'bottom',
            style: {
                fill: 'gray',
                'fill-opacity': 1
            }
        }
    ];
    return this.sub;
};

```

```
        }
    },
    {
        shape: new OG.RectangleShape('Sub2'),
        width: '50px',
        height: '50px',
        align: 'right',
        'vertical-align': 'top',
        style: {
            fill: 'gray',
            'fill-opacity': 1
        }
    },
    {
        shape: new OG.RectangleShape('Sub3'),
        width: '50px',
        height: '50px',
        top: '40px',
        left: '40px',
        style: {
            fill: 'gray',
            'fill-opacity': 1
        }
    },
    {
        shape: new OG.RectangleShape('Sub4'),
        width: '50px',
        height: '50px',
        right: '20%',
        bottom: '20%',
        style: {
            fill: 'gray',
            'fill-opacity': 1
        }
    },
    {
        shape: new OG.RectangleShape('Sub5'),
        width: '50px',
        height: '50px',
        right: '-150px',
        'vertical-align': 'middle',
        style: {
            fill: 'gray',
            'fill-opacity': 1
        }
    },
    {
        shape: new OG.RectangleShape('Sub6'),
        width: '50px',
        height: '50px',
        left: '-150px',
        'vertical-align': 'middle',
        style: {
            fill: 'gray',
            'fill-opacity': 1
        }
    }
];
```

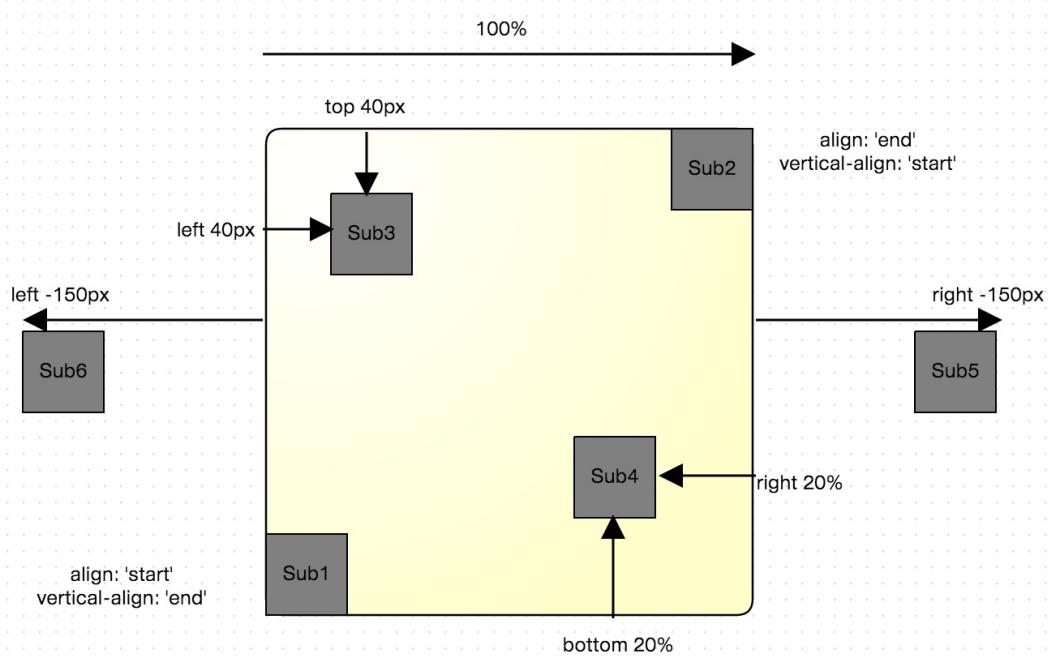
```

        style: {
            fill: 'gray',
            'fill-opacity': 1
        }
    };

    return this.sub;
};

canvas.drawShape([500, 300], new OG.SampleShape(), [300, 300]);

```



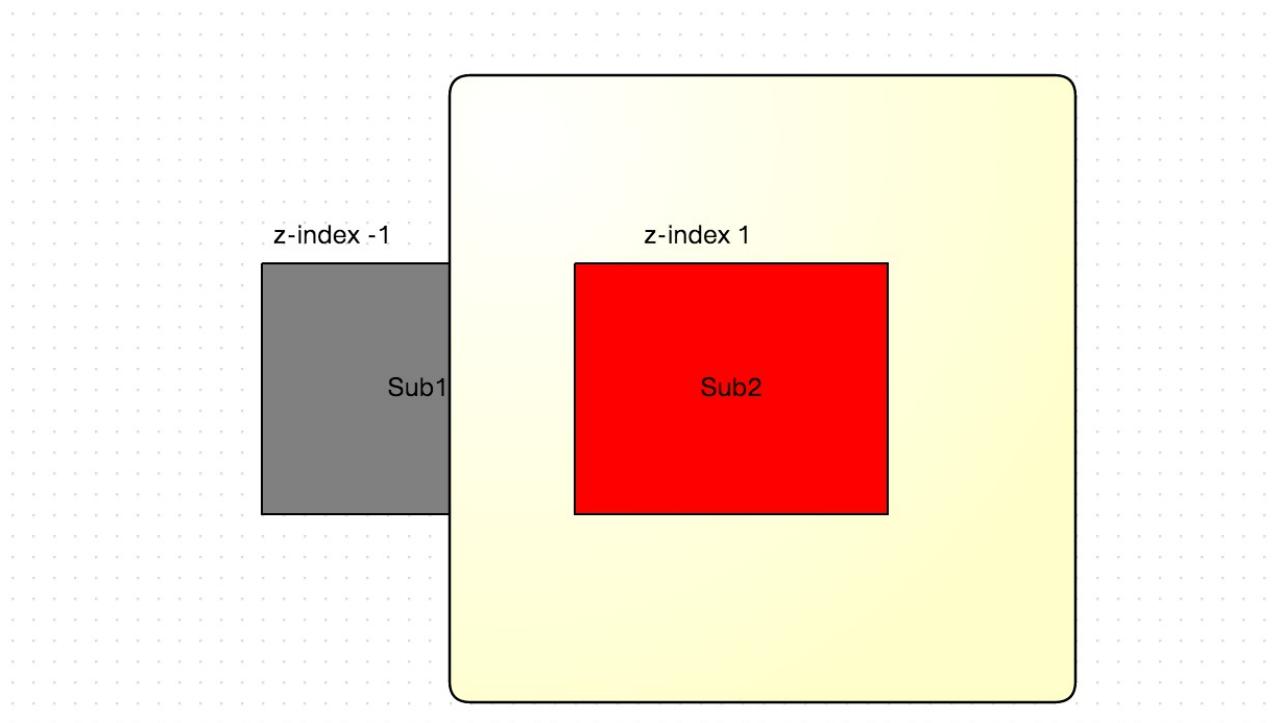
## Sub Shape index

z-index 프로퍼티로 서브 도형간의 앞,뒤 순서를 표현할 수 있습니다.

이 때, 음의 값이면 부모 도형의 뒤에, 양의 값이면 부모 도형의 앞에 위치하게 됩니다.

```
OG.shape.SampleShape.prototype.createSubShape = function () {
this.sub = [
{
    shape: new OG.RectangleShape('Sub1'),
    width: '50%',
    height: '40%',
    left: '-30%',
    'vertical-align' : 'middle',
    style: {
        'fill' : 'gray',
        'fill-opacity' : 1
    },
    'z-index': -1
},
{
    shape: new OG.RectangleShape('Sub2'),
    width: '50%',
    height: '40%',
    left: '20%',
    'vertical-align' : 'middle',
    style: {
        'fill' : 'red',
        'fill-opacity' : 1
    },
    'z-index': 1
}
];
return this.sub;
};

canvas.drawShape([500, 300], new OG.SampleShape(), [300, 300]);
```



## Sub Shape Data model

도형이 지니고 있는 데이터 값에 따라 서브 도형들이 유기적으로 대응할 수 있게 모델링 해 봅니다.

```
OG.shape.SampleShape.prototype.createShape = function () {
    if (this.geom) {
        return this.geom;
    }

    this.geom = new OG.geometry.Rectangle([0, 0], 100, 100);
    this.geom.style = new OG.geometry.Style({
        'fill-r': 1,
        'fill-cx': .1,
        'fill-cy': .1,
        "stroke-width": 1.2,
        fill: 'r(.1, .1)FFFFF-#FFFFCC',
        'fill-opacity': 1,
        r: '10'
    });
    return this.geom;
};

OG.shape.SampleShape.prototype.createSubShape = function () {
    this.sub = [];

    var loopShape;
    switch (this.data.LoopType) {
        case 'Standard' :
            loopShape = new
OG.ImageShape('resources/images/symbol/loop_standard.png');
            break;
```

```

        case 'MIParallel' :
            loopShape = new OG.MIParallel();
            break;
        case 'MISquential' :
            loopShape = new OG.MISquential();
            break;
    }
    if (loopShape) {
        this.sub.push({
            shape: loopShape,
            width: '15px',
            height: '15px',
            bottom: '5px',
            align: 'center',
            style: {}
        })
    }

var taskTypeShape;
switch (this.data.TaskType) {
    case "User":
        taskTypeShape = new
OG.ImageShape("resources/images/symbol/User.png");
        break;
    case "Send":
        taskTypeShape = new
OG.ImageShape('resources/images/symbol/Send.png');
        break;
    case "Receive":
        taskTypeShape = new
OG.ImageShape("resources/images/symbol/Receive.png");
        break;
    case "Manual":
        taskTypeShape = new
OG.ImageShape("resources/images/symbol/Manual.png");
        break;
    case "Service":
        taskTypeShape = new
OG.ImageShape("resources/images/symbol/Service.png");
        break;
    case "BusinessRule":
        taskTypeShape = new
OG.ImageShape("resources/images/symbol/BusinessRule.png");
        break;
    case "Script":
        taskTypeShape = new
OG.ImageShape("resources/images/symbol/Script.png");
        break;
    case "Mapper":
        taskTypeShape = new
OG.ImageShape("resources/images/symbol/Mapper.png");
        break;
}

```

```
        case "WebService":
            taskTypeShape = new
OG.ImageShape("resources/images/symbol/WebService.png");
            break;
    }
    if (taskTypeShape) {
        this.sub.push({
            shape: taskTypeShape,
            width: '20px',
            height: '20px',
            top: '5px',
            left: '5px',
            style: {}
        })
    }

var statusShape, statusAnimation;
switch (this.data.status) {
    case "Completed":
        statusShape = new
OG.ImageShape("resources/images/symbol/complete.png");
        break;
    case "Running":
        statusShape = new
OG.ImageShape('resources/images/symbol/running.png');
        statusAnimation = new OG.RectangleShape();
        break;
}
if (statusShape) {
    this.sub.push({
        shape: statusShape,
        width: '20px',
        height: '20px',
        right: '25px',
        top: '5px',
        style: {}
    })
}
if (statusAnimation) {
    this.sub.push({
        shape: statusAnimation,
        'z-index': -1,
        width: '120%',
        height: '120%',
        left: '-10%',
        top: '-10%',
        style: {
            'fill-opacity': 1,
            animation: [
                {
                    start: {
```

```

                fill: 'white'
            },
            to: {
                fill: '#C9E2FC'
            },
            ms: 1000
        },
        {
            start: {
                fill: '#C9E2FC'
            },
            to: {
                fill: 'white'
            },
            ms: 1000,
            delay: 1000
        }
    ],
    'animation-repeat': true,
    "fill": "#C9E2FC",
    "stroke-width": "0.2",
    "r": "10",
    'stroke-dasharray': '--'
}
})
}

return this.sub;
};

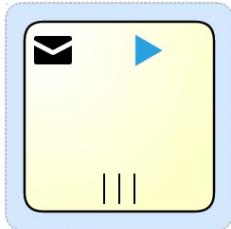
var sampleShape1 = new OG.SampleShape();
sampleShape1.data = {
    LoopType: 'Standard',
    TaskType: 'User',
    status : 'Completed'
};
canvas.drawShape([300, 300], sampleShape1, [100, 100]);

var sampleShape2 = new OG.SampleShape();
sampleShape2.data = {
    LoopType: 'MIParallel',
    TaskType: 'Send',
    status : 'Running'
};
canvas.drawShape([500, 300], sampleShape2, [100, 100]);

var sampleShape3 = new OG.SampleShape();
sampleShape3.data = {
    LoopType: 'MISquential',
    TaskType: 'Manual',
    status : 'Completed'
}

```

```
};  
canvas.drawShape([700, 300], sampleShape3, [100, 100]);
```



# Handler

핸들러 클래스는 오픈그래프에서 사용자가 ui 를 통해 발생시키는 이벤트 비지니스 로직을 담당하고 있으며, canvas.getEventHandler() 를 통해 얻을 수 있습니다.

```
var canvas = new OG.Canvas('canvas', [400, 400], 'white',
'url(resources/images/symbol/grid.gif)');
var eventHandler = canvas.getEventHandler();
```

핸들러 클래스의 api 를 직접 제어하거나 접근하는 것은 오픈그래프 개발자 영역이며, Api reference 문서의 EventHandler 파트를 통해 개발에 참조하시길 바랍니다.

# Renderer

렌더러 클래스는 오픈그래프의 실제 drawing 비지니스 로직을 담당하고 있으며, canvas.getRender() 를 통해 얻을 수 있습니다.

```
var canvas = new OG.Canvas('canvas', [400, 400], 'white',
'url(resources/images/symbol/grid.gif)');
var renderer = canvas.getRender();
```

렌더러 클래스의 api 를 직접 제어하거나 접근하는 것은 오픈그래프 개발자 영역이며,  
Api reference 문서의 IRenderer, RaphaelRenderer 파트를 통해 개발에 참조하시길 바랍니다.

# Objects

[OG](#) : object

## Functions

### [override\(origclass, overrides\)](#)

Adds a list of functions to the prototype of an existing class, overwriting any existing methods with the same name. Usage:

```
Ext.override(MyClass, {  
    newMethod1: function(){  
        // etc.  
    },  
    newMethod2: function(foo){  
        // etc.  
    }  
});
```

## OG : object

**Kind:** global namespace

- [OG](#) : object
  - [.common](#) : object
    - [.Constants](#)
      - [new OG.common.Constants\(\)](#)
      - [.GEOM\\_TYPE](#)
      - [.GEOM\\_NAME](#)
      - [.NUM\\_PRECISION](#)
      - [.NODE\\_TYPE](#)
      - [.SHAPE\\_TYPE](#)
      - [.EDGE\\_TYPE](#)
      - [.LABEL\\_SUFFIX](#)
      - [.LABEL\\_EDITOR\\_SUFFIX](#)
      - [.FROM\\_LABEL\\_SUFFIX](#)
      - [.TO\\_LABEL\\_SUFFIX](#)
      - [.RUBBER\\_BAND\\_ID](#)
      - [.RUBBER\\_BAND\\_TOLERANCE](#)
      - [.GUIDE\\_SUFFIX](#)
      - [.COLLAPSE\\_SUFFIX](#)
      - [.LOOPTYPE\\_SUFFIX](#)
      - [.TASKTYPE\\_SUFFIX](#)
      - [.INCLUSION\\_SUFFIX](#)
      - [.STATUS\\_SUFFIX](#)

- [.EXCEPTIONTYPE\\_SUFFIX](#)
- [.MOVE\\_SNAP\\_SIZE](#)
- [.DROP\\_OVER\\_BBOX\\_SUFFIX](#)
- [.TERMINAL\\_SUFFIX](#)
- [.TERMINAL](#)
- [.MARKER\\_TEMP\\_NODE](#)
- [.PATTERN\\_TEMP\\_NODE](#)
- [.MARKER\\_DEFS\\_SUFFIX](#)
- [.ORIGINAL\\_NODE](#)
- [.CONNECT\\_GUIDE\\_EVENT\\_AREA](#)
- [.CONNECT\\_GUIDE\\_SUFFIX](#)
  
- [.CurveUtil](#)
  - [new OG.common.CurveUtil\(\)](#)
  - [.CatmullRomSpline\(points\) ⇒ Object](#)
  - [.Bezier\(points\) ⇒ Object](#)
  - [.BSpline\(points, order\) ⇒ Object](#)
  
- [.HashMap](#)
  - [new OG.common.HashMap\(jsonObject\)](#)
  - [.map : Object](#)
  - [.put\(key, value\)](#)
  - [.get\(key\) ⇒ Object](#)
  - [.containsKey\(key\) ⇒ Boolean](#)
  - [.containsValue\(value\) ⇒ Boolean](#)
  - [.isEmpty\(\) ⇒ Boolean](#)
  - [.clear\(\)](#)
  - [.remove\(key\)](#)
  - [.keys\(\) ⇒ Array.<String>](#)
  - [.values\(\) ⇒ Array.<Object>](#)
  - [.size\(\) ⇒ Number](#)
  - [.toString\(\) ⇒ String](#)
  
- [.JSON](#)
  - [new OG.common.JSON\(\)](#)
  - [.encode ⇒ String](#)
  - [.decode ⇒ Object](#)
  - [.encodeDate\(d\) ⇒ String](#)
  
- [.Util](#)
  - [new OG.common.Util\(\)](#)
  - [.extend ⇒ function](#)
  - [.clone\(obj\) ⇒ Object](#)
  - [.round\(val\) ⇒ Number](#)
  - [.roundPrecision\(val, precision\) ⇒ Number](#)
  - [.roundGrid\(val, snapSize\) ⇒ Number](#)
  - [.apply\(obj, config, defaults\) ⇒ Object](#)
  
- [.geometry](#) : object

- [.BezierCurve](#)  $\Leftarrow$  [PolyLine](#)

- [new OG.geometry.BezierCurve\(controlPoints\)](#)
- [.controlPoints : Array.<Coordinate>](#)
- [.vertices : Array.<Coordinate>](#)
- [.TYPE : Number](#)
- [.IS CLOSED : Boolean](#)
- [.style : Style](#)
- [.boundary : Envelope](#)
- [.getControlPoints\(\) ⇒ Array.<Coordinate>](#)
- [.getVertices\(\) ⇒ Array.<Coordinate>](#)
- [.move\(offsetX, offsetY\) ⇒ Geometry](#)
- [.resize\(upper, lower, left, right\) ⇒ Geometry](#)
- [.rotate\(angle, origin\) ⇒ Geometry](#)
- [.toString\(\) ⇒ String](#)
- [.angleBetweenPoints\(prev, next\) ⇒ Number](#)
- [.isRightAngleBetweenPoints\(prev, next\) ⇒ Object](#)
- [.angleBetweenThreePoints\(prev, next\) ⇒ Number](#)
- [.isEqual\( geometry \) ⇒ Boolean](#)
- [.contains\( geometry \) ⇒ Boolean](#)
- [.isWithin\( geometry \) ⇒ Boolean](#)
- [.getBoundary\(\) ⇒ Envelope](#)
- [.getCentroid\(\) ⇒ Coordinate](#)
- [.minDistance\( coordinate \) ⇒ Number](#)
- [.distance\( geometry \) ⇒ Number](#)
- [.getLength\(\) ⇒ Number](#)
- [.moveCentroid\(重心\)](#)
- [.resizeBox\(width, height\) ⇒ Geometry](#)
- [.fitToBoundary\(envelope\) ⇒ Geometry](#)
- [.convertCoordinate\(coordinate\) ⇒ Coordinate](#)
- [.distanceToLine\(p, line\) ⇒ Number](#)
- [.distanceLineToLine\(line1, line2\) ⇒ Number](#)
- [.intersectToLine\(line\) ⇒ Array.<Coordinate>](#)
- [.shortestIntersectToLine\(line\) ⇒ Array.<Coordinate>](#)
- [.intersectLineToLine\(line1, line2, extension\) ⇒ Coordinate](#)
- [.intersectCircleToLine\(center, radius, from, to\) ⇒ Array.<Coordinate>](#)
- [.intersectPointToLine\(p, line\) ⇒ Coordinate](#)
- [.getPercentageDistanceFromPoint\( coordinate \) ⇒ Object](#)
- [.containsPoint\( coordinate \) ⇒ boolean](#)
- [.getPointFromPercentageDistance\(pXpY\) ⇒ Coordinate](#)
- [.getParallelLine\(from, to, distance\) ⇒ Array.<Coordinate>](#)
- [.getParallelPath\(line, distance\)](#)
- [.reset\(\)](#)

- [.Circle](#)  $\Leftarrow$  [Ellipse](#)

- [new OG.geometry.Circle\(center, radius\)](#)
- [.vertices : Array.<Coordinate>](#)

- [.TYPE](#) : Number
- [.IS\\_CLOSED](#) : Boolean
- [.style](#) : [Style](#)
- [.boundary](#) : [Envelope](#)
- [.getVertices\(\)](#) ⇒ [Array.<Coordinate>](#)
- [.getControlPoints\(\)](#) ⇒ [Array.<Coordinate>](#)
- [.getLength\(\)](#) ⇒ Number
- [.toString\(\)](#) ⇒ String
- [.move\(offsetX, offsetY\)](#) ⇒ [Geometry](#)
- [.resize\(upper, lower, left, right\)](#) ⇒ [Geometry](#)
- [.rotate\(angle, origin\)](#) ⇒ [Geometry](#)
- [.angleBetweenPoints\(prev, next\)](#) ⇒ Number
- [.isRightAngleBetweenPoints\(prev, next\)](#) ⇒ Object
- [.angleBetweenThreePoints\(prev, next\)](#) ⇒ Number
- [.isEquals\( geometry\)](#) ⇒ Boolean
- [.isContains\( geometry\)](#) ⇒ Boolean
- [.isWithin\( geometry\)](#) ⇒ Boolean
- [.getBoundary\(\)](#) ⇒ [Envelope](#)
- [.getCentroid\(\)](#) ⇒ [Coordinate](#)
- [.minDistance\( coordinate\)](#) ⇒ Number
- [.distance\( geometry\)](#) ⇒ Number
- [.moveCentroid\(중심\)](#)
- [.resizeBox\(width, height\)](#) ⇒ [Geometry](#)
- [.fitToBoundary\(envelope\)](#) ⇒ [Geometry](#)
- [.convertCoordinate\(coordinate\)](#) ⇒ [Coordinate](#)
- [.distanceToLine\(p, line\)](#) ⇒ Number
- [.distanceLineToLine\(line1, line2\)](#) ⇒ Number
- [.intersectToLine\(line\)](#) ⇒ [Array.<Coordinate>](#)
- [.shortestIntersectToLine\(line\)](#) ⇒ [Array.<Coordinate>](#)
- [.intersectLineToLine\(line1, line2, extension\)](#) ⇒ [Coordinate](#)
- [.intersectCircleToLine\(center, radius, from, to\)](#) ⇒ [Array.<Coordinate>](#)
- [.intersectPointToLine\(p, line\)](#) ⇒ [Coordinate](#)
- [.getPercentageDistanceFromPoint\( coordinate\)](#) ⇒ Object
- [.isContainsPoint\( coordinate\)](#) ⇒ boolean
- [.getPointFromPercentageDistance\(pXpY\)](#) ⇒ [Coordinate](#)
- [.getParallelLine\(from, to, distance\)](#) ⇒ [Array.<Coordinate>](#)
- [.getParallelPath\(line, distance\)](#)
- [.reset\(\)](#)
  
- [.Coordinate](#)
  - [new OG.geometry.Coordinate\(x, y\)](#)
  - [.x](#) : Number
  - [.y](#) : Number
  - [.distance\(coordinate\)](#) ⇒ Number
  - [.move\(offsetX, offsetY\)](#) ⇒ [Coordinate](#)
  - [.rotate\(angle, origin\)](#) ⇒ [Coordinate](#)
  - [.isEquals\(coordinate\)](#) ⇒ Boolean

- [.toString\(\)](#) ⇒ String
- [.Curve](#) ⇐ [PolyLine](#)
  - [new OG.geometry.Curve\(controlPoints\)](#)
  - [.vertices : Array.<Coordinate>](#)
  - [.TYPE : Number](#)
  - [.IS CLOSED : Boolean](#)
  - [.style : Style](#)
  - [.boundary : Envelope](#)
  - [.getControlPoints\(\) ⇒ Array.<Coordinate>](#)
  - [.getVertices\(\) ⇒ Array.<Coordinate>](#)
  - [.move\(offsetX, offsetY\) ⇒ Geometry](#)
  - [.resize\(upper, lower, left, right\) ⇒ Geometry](#)
  - [.rotate\(angle, origin\) ⇒ Geometry](#)
  - [.toString\(\) ⇒ String](#)
  - [.angleBetweenPoints\(prev, next\) ⇒ Number](#)
  - [.isRightAngleBetweenPoints\(prev, next\) ⇒ Object](#)
  - [.angleBetweenThreePoints\(prev, next\) ⇒ Number](#)
  - [.isEqual\( geometry\) ⇒ Boolean](#)
  - [.contains\( geometry\) ⇒ Boolean](#)
  - [.within\( geometry\) ⇒ Boolean](#)
  - [.getBoundary\(\) ⇒ Envelope](#)
  - [.getCentroid\(\) ⇒ Coordinate](#)
  - [.minDistance\( coordinate\) ⇒ Number](#)
  - [.distance\( geometry\) ⇒ Number](#)
  - [.getLength\(\) ⇒ Number](#)
  - [.moveCentroid\(중심\)](#)
  - [.resizeBox\(width, height\) ⇒ Geometry](#)
  - [.fitToBoundary\(envelope\) ⇒ Geometry](#)
  - [.convertCoordinate\(coordinate\) ⇒ Coordinate](#)
  - [.distanceToLine\(p, line\) ⇒ Number](#)
  - [.distanceLineToLine\(line1, line2\) ⇒ Number](#)
  - [.intersectToLine\(line\) ⇒ Array.<Coordinate>](#)
  - [.shortestIntersectToLine\(line\) ⇒ Array.<Coordinate>](#)
  - [.intersectLineToLine\(line1, line2, extension\) ⇒ Coordinate](#)
  - [.intersectCircleToLine\(center, radius, from, to\) ⇒ Array.<Coordinate>](#)
  - [.intersectPointToLine\(p, line\) ⇒ Coordinate](#)
  - [.getPercentageDistanceFromPoint\( coordinate\) ⇒ Object](#)
  - [.containsPoint\( coordinate\) ⇒ boolean](#)
  - [.getPointFromPercentageDistance\(pXpY\) ⇒ Coordinate](#)
  - [.getParallelLine\(from, to, distance\) ⇒ Array.<Coordinate>](#)
  - [.getParallelPath\(line, distance\)](#)
  - [.reset\(\)](#)
- [.Ellipse](#) ⇐ [Curve](#)
  - [new OG.geometry.Ellipse\(center, radiusX, radiusY, angle\)](#)
  - [.vertices : Array.<Coordinate>](#)

- [.TYPE](#) : Number
- [.IS\\_CLOSED](#) : Boolean
- [.style](#) : Style
- [.boundary](#) : Envelope
- [.getControlPoints\(\)](#) ⇒ Array.<Coordinate>
- [.getVertices\(\)](#) ⇒ Array.<Coordinate>
- [.toString\(\)](#) ⇒ String
- [.move\(offsetX, offsetY\)](#) ⇒ Geometry
- [.resize\(upper, lower, left, right\)](#) ⇒ Geometry
- [.rotate\(angle, origin\)](#) ⇒ Geometry
- [.angleBetweenPoints\(prev, next\)](#) ⇒ Number
- [.isRightAngleBetweenPoints\(prev, next\)](#) ⇒ Object
- [.angleBetweenThreePoints\(prev, next\)](#) ⇒ Number
- [.isEquals\( geometry\)](#) ⇒ Boolean
- [.isContains\( geometry\)](#) ⇒ Boolean
- [.isWithin\( geometry\)](#) ⇒ Boolean
- [.getBoundary\(\)](#) ⇒ Envelope
- [.getCentroid\(\)](#) ⇒ Coordinate
- [.minDistance\( coordinate\)](#) ⇒ Number
- [.distance\( geometry\)](#) ⇒ Number
- [.getLength\(\)](#) ⇒ Number
- [.moveCentroid\(중심\)](#)
- [.resizeBox\(width, height\)](#) ⇒ Geometry
- [.fitToBoundary\(envelope\)](#) ⇒ Geometry
- [.convertCoordinate\(coordinate\)](#) ⇒ Coordinate
- [.distanceToLine\(p, line\)](#) ⇒ Number
- [.distanceLineToLine\(line1, line2\)](#) ⇒ Number
- [.intersectToLine\(line\)](#) ⇒ Array.<Coordinate>
- [.shortestIntersectToLine\(line\)](#) ⇒ Array.<Coordinate>
- [.intersectLineToLine\(line1, line2, extension\)](#) ⇒ Coordinate
- [.intersectCircleToLine\(center, radius, from, to\)](#) ⇒ Array.<Coordinate>
- [.intersectPointToLine\(p, line\)](#) ⇒ Coordinate
- [.getPercentageDistanceFromPoint\( coordinate\)](#) ⇒ Object
- [.isContainsPoint\( coordinate\)](#) ⇒ boolean
- [.getPointFromPercentageDistance\(pXpY\)](#) ⇒ Coordinate
- [.getParallelLine\(from, to, distance\)](#) ⇒ Array.<Coordinate>
- [.getParallelPath\(line, distance\)](#)
- [.reset\(\)](#)
  
- [.Envelope](#)
  - [new OG.geometry.Envelope\(upperLeft, width, height\)](#)
  - [.getUpperLeft\(\)](#) ⇒ Coordinate
  - [.setUpperLeft\(upperLeft\)](#)
  - [.getUpperRight\(\)](#) ⇒ Coordinate
  - [.getLowerRight\(\)](#) ⇒ Coordinate
  - [.getLowerLeft\(\)](#) ⇒ Coordinate
  - [.getLeftCenter\(\)](#) ⇒ Coordinate

- [.getUpperCenter\(\) ⇒ Coordinate](#)
- [.getRightCenter\(\) ⇒ Coordinate](#)
- [.getLowerCenter\(\) ⇒ Coordinate](#)
- [.getCentroid\(\) ⇒ Coordinate](#)
- [.setCentroid\(centroid\)](#)
- [.getWidth\(\) ⇒ Number](#)
- [.setWidth\(width\)](#)
- [.getHeight\(\) ⇒ Number](#)
- [.setHeight\(height\)](#)
- [.getVertices\(\) ⇒ Array.<Coordinate>](#)
- [.isContains\(coordinate\) ⇒ Boolean](#)
- [.isContainsAll\(coordinateArray\) ⇒ Boolean](#)
- [.getHowManyContains\(coordinateArray\) ⇒ Boolean](#)
- [.isContainsOnce\(coordinateArray\) ⇒ Boolean](#)
- [.move\(offsetX, offsetY\) ⇒ Envelope](#)
- [.resize\(upper, lower, left, right\) ⇒ Envelope](#)
- [.isEquals\(Envelope\) ⇒ Boolean](#)
- [.toString\(\) ⇒ String](#)
  
- [.Geometry](#)
  - [new OG.geometry.Geometry\(\)](#)
  - [.TYPE : Number](#)
  - [.IS\\_CLOSED : Boolean](#)
  - [.style : Style](#)
  - [.boundary : Envelope](#)
  - [.isEquals\( geometry\) ⇒ Boolean](#)
  - [.isContains\( geometry\) ⇒ Boolean](#)
  - [.isWithin\( geometry\) ⇒ Boolean](#)
  - [.getBoundary\(\) ⇒ Envelope](#)
  - [.getCentroid\(\) ⇒ Coordinate](#)
  - [.getVertices\(\) ⇒ Array.<Coordinate>](#)
  - [.minDistance\( coordinate\) ⇒ Number](#)
  - [.distance\( geometry\) ⇒ Number](#)
  - [.getLength\(\) ⇒ Number](#)
  - [.move\(offsetX, offsetY\) ⇒ Geometry](#)
  - [.moveCentroid\(중심\)](#)
  - [.resize\(upper, lower, left, right\) ⇒ Geometry](#)
  - [.resizeBox\(width, height\) ⇒ Geometry](#)
  - [.rotate\(angle, origin\) ⇒ Geometry](#)
  - [.fitToBoundary\(envelope\) ⇒ Geometry](#)
  - [.convertCoordinate\(coordinate\) ⇒ Coordinate](#)
  - [.distanceToLine\(p, line\) ⇒ Number](#)
  - [.distanceLineToLine\(line1, line2\) ⇒ Number](#)
  - [.intersectToLine\(line\) ⇒ Array.<Coordinate>](#)
  - [.shortestIntersectToLine\(line\) ⇒ Array.<Coordinate>](#)
  - [.intersectLineToLine\(line1, line2, extension\) ⇒ Coordinate](#)
  - [.intersectCircleToLine\(center, radius, from, to\) ⇒ Array.<Coordinate>](#)

- [.intersectPointToLine\(p, line\) ⇒ Coordinate](#)
- [.getPercentageDistanceFromPoint\( coordinate\) ⇒ Object](#)
- [.isContainsPoint\( coordinate\) ⇒ boolean](#)
- [.getPointFromPercentageDistance\(pXpY\) ⇒ Coordinate](#)
- [.getParallelLine\(from, to, distance\) ⇒ Array.<Coordinate>](#)
- [.getParallelPath\(line, distance\)](#)
- [.reset\(\)](#)
  
- [\*\*.GeometryCollection\*\* ← Geometry](#)
  - [new OG.geometry.GeometryCollection\(geometries\)](#)
  - [.geometries : Array.<Geometry>](#)
  - [.TYPE : Number](#)
  - [.IS CLOSED : Boolean](#)
  - [.style : Style](#)
  - [.boundary : Envelope](#)
  - [.toString\(\) ⇒ String](#)
  - [.isEqual\( geometry\) ⇒ Boolean](#)
  - [.isContains\( geometry\) ⇒ Boolean](#)
  - [.isWithin\( geometry\) ⇒ Boolean](#)
  - [.getBoundary\(\) ⇒ Envelope](#)
  - [.getCentroid\(\) ⇒ Coordinate](#)
  - [.getVertices\(\) ⇒ Array.<Coordinate>](#)
  - [.minDistance\( coordinate\) ⇒ Number](#)
  - [.distance\( geometry\) ⇒ Number](#)
  - [.getLength\(\) ⇒ Number](#)
  - [.move\(offsetX, offsetY\) ⇒ Geometry](#)
  - [.moveCentroid\( 중심\)](#)
  - [.resize\(upper, lower, left, right\) ⇒ Geometry](#)
  - [.resizeBox\(width, height\) ⇒ Geometry](#)
  - [.rotate\(angle, origin\) ⇒ Geometry](#)
  - [.fitToBoundary\(envelope\) ⇒ Geometry](#)
  - [.convertCoordinate\(coordinate\) ⇒ Coordinate](#)
  - [.distanceToLine\(p, line\) ⇒ Number](#)
  - [.distanceLineToLine\(line1, line2\) ⇒ Number](#)
  - [.intersectToLine\(line\) ⇒ Array.<Coordinate>](#)
  - [.shortestIntersectToLine\(line\) ⇒ Array.<Coordinate>](#)
  - [.intersectLineToLine\(line1, line2, extension\) ⇒ Coordinate](#)
  - [.intersectCircleToLine\(center, radius, from, to\) ⇒ Array.<Coordinate>](#)
  - [.intersectPointToLine\(p, line\) ⇒ Coordinate](#)
  - [.getPercentageDistanceFromPoint\( coordinate\) ⇒ Object](#)
  - [.isContainsPoint\( coordinate\) ⇒ boolean](#)
  - [.getPointFromPercentageDistance\(pXpY\) ⇒ Coordinate](#)
  - [.getParallelLine\(from, to, distance\) ⇒ Array.<Coordinate>](#)
  - [.getParallelPath\(line, distance\)](#)
  - [.reset\(\)](#)
  
- [\*\*.Line\*\* ← PolyLine](#)

- [new OG.geometry.Line\(from, to\)](#)
- [.vertices : Array.<Coordinate>](#)
- [.TYPE : Number](#)
- [.IS CLOSED : Boolean](#)
- [.style : Style](#)
- [.boundary : Envelope](#)
- [.getVertices\(\) ⇒ Array.<Coordinate>](#)
- [.move\(offsetX, offsetY\) ⇒ Geometry](#)
- [.resize\(upper, lower, left, right\) ⇒ Geometry](#)
- [.rotate\(angle, origin\) ⇒ Geometry](#)
- [.toString\(\) ⇒ String](#)
- [.angleBetweenPoints\(prev, next\) ⇒ Number](#)
- [.isRightAngleBetweenPoints\(prev, next\) ⇒ Object](#)
- [.angleBetweenThreePoints\(prev, next\) ⇒ Number](#)
- [.isEquals\( geometry\) ⇒ Boolean](#)
- [.isContains\( geometry\) ⇒ Boolean](#)
- [.isWithin\( geometry\) ⇒ Boolean](#)
- [.getBoundary\(\) ⇒ Envelope](#)
- [.getCentroid\(\) ⇒ Coordinate](#)
- [.minDistance\( coordinate\) ⇒ Number](#)
- [.distance\( geometry\) ⇒ Number](#)
- [.getLength\(\) ⇒ Number](#)
- [.moveCentroid\( 중심\)](#)
- [.resizeBox\(width, height\) ⇒ Geometry](#)
- [.fitToBoundary\(envelope\) ⇒ Geometry](#)
- [.convertCoordinate\(coordinate\) ⇒ Coordinate](#)
- [.distanceToLine\(p, line\) ⇒ Number](#)
- [.distanceLineToLine\(line1, line2\) ⇒ Number](#)
- [.intersectToLine\(line\) ⇒ Array.<Coordinate>](#)
- [.shortestIntersectToLine\(line\) ⇒ Array.<Coordinate>](#)
- [.intersectLineToLine\(line1, line2, extension\) ⇒ Coordinate](#)
- [.intersectCircleToLine\(center, radius, from, to\) ⇒ Array.<Coordinate>](#)
- [.intersectPointToLine\(p, line\) ⇒ Coordinate](#)
- [.getPercentageDistanceFromPoint\( coordinate\) ⇒ Object](#)
- [.isContainsPoint\( coordinate\) ⇒ boolean](#)
- [.getPointFromPercentageDistance\(pXpY\) ⇒ Coordinate](#)
- [.getParallelLine\(from, to, distance\) ⇒ Array.<Coordinate>](#)
- [.getParallelPath\(line, distance\)](#)
- [.reset\(\)](#)
  
- [.Point ⇐ Geometry](#)
  - [new OG.geometry.Point\(coordinate\)](#)
  - [.coordinate : Coordinate](#)
  - [.vertices : Array.<Coordinate>](#)
  - [.TYPE : Number](#)
  - [.IS CLOSED : Boolean](#)
  - [.style : Style](#)

- [.boundary : Envelope](#)
- [.toString\(\) ⇒ String](#)
- [.isEquals\( geometry\) ⇒ Boolean](#)
- [.isContains\( geometry\) ⇒ Boolean](#)
- [.isWithin\( geometry\) ⇒ Boolean](#)
- [.getBoundary\(\) ⇒ Envelope](#)
- [.getCentroid\(\) ⇒ Coordinate](#)
- [.getVertices\(\) ⇒ Array.<Coordinate>](#)
- [.minDistance\( coordinate\) ⇒ Number](#)
- [.distance\( geometry\) ⇒ Number](#)
- [.getLength\(\) ⇒ Number](#)
- [.move\(offsetX, offsetY\) ⇒ Geometry](#)
- [.moveCentroid\(중심\)](#)
- [.resize\(upper, lower, left, right\) ⇒ Geometry](#)
- [.resizeBox\(width, height\) ⇒ Geometry](#)
- [.rotate\(angle, origin\) ⇒ Geometry](#)
- [.fitToBoundary\(envelope\) ⇒ Geometry](#)
- [.convertCoordinate\(coordinate\) ⇒ Coordinate](#)
- [.distanceToLine\(p, line\) ⇒ Number](#)
- [.distanceLineToLine\(line1, line2\) ⇒ Number](#)
- [.intersectToLine\(line\) ⇒ Array.<Coordinate>](#)
- [.shortestIntersectToLine\(line\) ⇒ Array.<Coordinate>](#)
- [.intersectLineToLine\(line1, line2, extension\) ⇒ Coordinate](#)
- [.intersectCircleToLine\(center, radius, from, to\) ⇒ Array.<Coordinate>](#)
- [.intersectPointToLine\(p, line\) ⇒ Coordinate](#)
- [.getPercentageDistanceFromPoint\( coordinate\) ⇒ Object](#)
- [.isContainsPoint\( coordinate\) ⇒ boolean](#)
- [.getPointFromPercentageDistance\(pXpY\) ⇒ Coordinate](#)
- [.getParallelLine\(from, to, distance\) ⇒ Array.<Coordinate>](#)
- [.getParallelPath\(line, distance\)](#)
- [.reset\(\)](#)
  
- [.PolyLine ⇐ Geometry](#)
  - [new OG.geometry.PolyLine\(vertices\)](#)
  - [.vertices : Array.<Coordinate>](#)
  - [.TYPE : Number](#)
  - [.IS\\_CLOSED : Boolean](#)
  - [.style : Style](#)
  - [.boundary : Envelope](#)
  - [.toString\(\) ⇒ String](#)
  - [.angleBetweenPoints\(prev, next\) ⇒ Number](#)
  - [.isRightAngleBetweenPoints\(prev, next\) ⇒ Object](#)
  - [.angleBetweenThreePoints\(prev, next\) ⇒ Number](#)
  - [.isEquals\( geometry\) ⇒ Boolean](#)
  - [.isContains\( geometry\) ⇒ Boolean](#)
  - [.isWithin\( geometry\) ⇒ Boolean](#)
  - [.getBoundary\(\) ⇒ Envelope](#)

- [.getCentroid\(\) ⇒ Coordinate](#)
- [.getVertices\(\) ⇒ Array.<Coordinate>](#)
- [.minDistance\( coordinate\) ⇒ Number](#)
- [.distance\( geometry\) ⇒ Number](#)
- [.getLength\(\) ⇒ Number](#)
- [.move\(offsetX, offsetY\) ⇒ Geometry](#)
- [.moveCentroid\(중심\)](#)
- [.resize\(upper, lower, left, right\) ⇒ Geometry](#)
- [.resizeBox\(width, height\) ⇒ Geometry](#)
- [.rotate\(angle, origin\) ⇒ Geometry](#)
- [.fitToBoundary\(envelope\) ⇒ Geometry](#)
- [.convertCoordinate\(coordinate\) ⇒ Coordinate](#)
- [.distanceToLine\(p, line\) ⇒ Number](#)
- [.distanceLineToLine\(line1, line2\) ⇒ Number](#)
- [.intersectToLine\(line\) ⇒ Array.<Coordinate>](#)
- [.shortestIntersectToLine\(line\) ⇒ Array.<Coordinate>](#)
- [.intersectLineToLine\(line1, line2, extension\) ⇒ Coordinate](#)
- [.intersectCircleToLine\(center, radius, from, to\) ⇒ Array.<Coordinate>](#)
- [.intersectPointToLine\(p, line\) ⇒ Coordinate](#)
- [.getPercentageDistanceFromPoint\( coordinate\) ⇒ Object](#)
- [.isContainsPoint\( coordinate\) ⇒ boolean](#)
- [.getPointFromPercentageDistance\(pXpY\) ⇒ Coordinate](#)
- [.getParallelLine\(from, to, distance\) ⇒ Array.<Coordinate>](#)
- [.getParallelPath\(line, distance\)](#)
- [.reset\(\)](#)
  
- [\*\*Polygon\*\* ⇌ PolyLine](#)
  - [new OG.geometry.Polygon\(vertices\)](#)
  - [.vertices : Array.<Coordinate>](#)
  - [.TYPE : Number](#)
  - [.IS CLOSED : Boolean](#)
  - [.style : Style](#)
  - [.boundary : Envelope](#)
  - [.getVertices\(\) ⇒ Array.<Coordinate>](#)
  - [.move\(offsetX, offsetY\) ⇒ Geometry](#)
  - [.resize\(upper, lower, left, right\) ⇒ Geometry](#)
  - [.rotate\(angle, origin\) ⇒ Geometry](#)
  - [.toString\(\) ⇒ String](#)
  - [.angleBetweenPoints\(prev, next\) ⇒ Number](#)
  - [.isRightAngleBetweenPoints\(prev, next\) ⇒ Object](#)
  - [.angleBetweenThreePoints\(prev, next\) ⇒ Number](#)
  - [.isEquals\( geometry\) ⇒ Boolean](#)
  - [.isContains\( geometry\) ⇒ Boolean](#)
  - [.isWithin\( geometry\) ⇒ Boolean](#)
  - [.getBoundary\(\) ⇒ Envelope](#)
  - [.getCentroid\(\) ⇒ Coordinate](#)
  - [.minDistance\( coordinate\) ⇒ Number](#)

- [.distance\( geometry \)](#) ⇒ Number
- [.getLength\(\)](#) ⇒ Number
- [.moveCentroid\(중심\)](#)
- [.resizeBox\(width, height\)](#) ⇒ [Geometry](#)
- [.fitToBoundary\(envelope\)](#) ⇒ [Geometry](#)
- [.convertCoordinate\(coordinate\)](#) ⇒ [Coordinate](#)
- [.distanceToLine\(p, line\)](#) ⇒ Number
- [.distanceLineToLine\(line1, line2\)](#) ⇒ Number
- [.intersectToLine\(line\)](#) ⇒ [Array.<Coordinate>](#)
- [.shortestIntersectToLine\(line\)](#) ⇒ [Array.<Coordinate>](#)
- [.intersectLineToLine\(line1, line2, extension\)](#) ⇒ [Coordinate](#)
- [.intersectCircleToLine\(center, radius, from, to\)](#) ⇒ [Array.<Coordinate>](#)
- [.intersectPointToLine\(p, line\)](#) ⇒ [Coordinate](#)
- [.getPercentageDistanceFromPoint\( coordinate \)](#) ⇒ Object
- [.isContainsPoint\( coordinate \)](#) ⇒ boolean
- [.getPointFromPercentageDistance\(pXpY\)](#) ⇒ [Coordinate](#)
- [.getParallelLine\(from, to, distance\)](#) ⇒ [Array.<Coordinate>](#)
- [.getParallelPath\(line, distance\)](#)
- [.reset\(\)](#)
  
- [.Rectangle ⇌ Polygon](#)
  - [new OG.geometry.Rectangle\(upperLeft, width, height\)](#)
  - [.vertices : Array.<Coordinate>](#)
  - [.TYPE : Number](#)
  - [.IS CLOSED : Boolean](#)
  - [.style : Style](#)
  - [.boundary : Envelope](#)
  - [.getVertices\(\)](#) ⇒ [Array.<Coordinate>](#)
  - [.move\(offsetX, offsetY\)](#) ⇒ [Geometry](#)
  - [.resize\(upper, lower, left, right\)](#) ⇒ [Geometry](#)
  - [.rotate\(angle, origin\)](#) ⇒ [Geometry](#)
  - [.toString\(\)](#) ⇒ String
  - [.angleBetweenPoints\(prev, next\)](#) ⇒ Number
  - [.isRightAngleBetweenPoints\(prev, next\)](#) ⇒ Object
  - [.angleBetweenThreePoints\(prev, next\)](#) ⇒ Number
  - [.isEquals\( geometry \)](#) ⇒ Boolean
  - [.isContains\( geometry \)](#) ⇒ Boolean
  - [.isWithin\( geometry \)](#) ⇒ Boolean
  - [.getBoundary\(\)](#) ⇒ [Envelope](#)
  - [.getCentroid\(\)](#) ⇒ [Coordinate](#)
  - [.minDistance\( coordinate \)](#) ⇒ Number
  - [.distance\( geometry \)](#) ⇒ Number
  - [.getLength\(\)](#) ⇒ Number
  - [.moveCentroid\(중심\)](#)
  - [.resizeBox\(width, height\)](#) ⇒ [Geometry](#)
  - [.fitToBoundary\(envelope\)](#) ⇒ [Geometry](#)
  - [.convertCoordinate\(coordinate\)](#) ⇒ [Coordinate](#)

- [.distanceToLine\(p, line\) ⇒ Number](#)
- [.distanceLineToLine\(line1, line2\) ⇒ Number](#)
- [.intersectToLine\(line\) ⇒ Array.<Coordinate>](#)
- [.shortestIntersectToLine\(line\) ⇒ Array.<Coordinate>](#)
- [.intersectLineToLine\(line1, line2, extension\) ⇒ Coordinate](#)
- [.intersectCircleToLine\(center, radius, from, to\) ⇒ Array.<Coordinate>](#)
- [.intersectPointToLine\(p, line\) ⇒ Coordinate](#)
- [.getPercentageDistanceFromPoint\( coordinate\) ⇒ Object](#)
- [.isContainsPoint\( coordinate\) ⇒ boolean](#)
- [.getPointFromPercentageDistance\(pXpY\) ⇒ Coordinate](#)
- [.getParallelLine\(from, to, distance\) ⇒ Array.<Coordinate>](#)
- [.getParallelPath\(line, distance\)](#)
- [.reset\(\)](#)
  
- [.Style ⇐ HashMap](#)
  - [new OG.geometry.Style\(style\)](#)
  - [.map : Object](#)
  - [.put\(key, value\)](#)
  - [.get\(key\) ⇒ Object](#)
  - [.containsKey\(key\) ⇒ Boolean](#)
  - [.containsValue\(value\) ⇒ Boolean](#)
  - [.isEmpty\(\) ⇒ Boolean](#)
  - [.clear\(\)](#)
  - [.remove\(key\)](#)
  - [.keys\(\) ⇒ Array.<String>](#)
  - [.values\(\) ⇒ Array.<Object>](#)
  - [.size\(\) ⇒ Number](#)
  - [.toString\(\) ⇒ String](#)
  
- [.graph : object](#)
  - [.Canvas](#)
    - [new OG.graph.Canvas\(container, containerSize, backgroundColor, backgroundImage\)](#)
    - [.initConfig\(config\)](#)
    - [.getRenderer\(\) ⇒ OG.RaphaelRenderer](#)
    - [.getContainer\(\) ⇒ HTMLElement](#)
    - [.getEventHandler\(\) ⇒ OG.EventHandler](#)
    - [.addSlider\(\)](#)
    - [.removeSlider\(\)](#)
    - [.drawShape\(position, shape, size, style, id, parentId, preventEvent\) ⇒ Element](#)
    - [.drawTransformer\(position, label, inputs, outputs, id\) ⇒ Element](#)
    - [.setShapeStyle\(shapeElement, style\)](#)
    - [.setTextListInController\(shapeElement, textList\)](#)
    - [.getTextListInController\(shapeElement\)](#)
    - [.drawLabel\(shapeElement, text, style\) ⇒ Element](#)
    - [.redrawConnectedEdge\(element\)](#)

- [.reconnect\(edge\)](#) ⇒ Element
- [.connect\(fromElement, toElement, style, label, fromP, toP, preventTrigger, id, edgeShape\)](#) ⇒ \* | Element
- [.connectWithTerminalId\(fromTerminal, toTerminal, style, label\)](#) ⇒ String | String | [geometry](#)
- [.disconnect\(element\)](#)
- [.group\(elements\)](#) ⇒ Element
- [.ungroup\(groupElements\)](#) ⇒ Array.<Element>
- [.addToGroup\(groupElement, elements\)](#)
- [.collapse\(element\)](#)
- [.expand\(element\)](#)
- [.clear\(\)](#)
- [.removeShape\(element\)](#)
- [.removeChild\(element\)](#)
- [.removeGuide\(element\)](#)
- [.removeAllGuide\(\)](#)
- [.getRootElement\(\)](#) ⇒ Element
- [.getRootGroup\(\)](#) ⇒ Element
- [.getElementByPoint\(position\)](#) ⇒ Element
- [.getElementsByBBox\(envelope\)](#) ⇒ Array.<Element>
- [.setAttr\(element, attribute\)](#)
- [.getAttr\(element, attrName\)](#) ⇒ Object
- [.toFront\(element\)](#)
- [.toBack\(element\)](#)
- [.bringForward\(element\)](#)
- [.sendBackward\(element\)](#)
- [.getCanvasSize\(\)](#) ⇒ Array.<Number>
- [.setCanvasSize\(size\)](#)
- [.fitCanvasSize\(minSize, fitScale\)](#)
- [.setViewBox\(position, size, isFit\)](#)
- [.getScale\(\)](#) ⇒ Number
- [.setScale\(scale\)](#)
- [.show\(element\)](#)
- [.hide\(element\)](#)
- [.appendChild\(srcElement, targetElement\)](#) ⇒ Element
- [.insertAfter\(srcElement, targetElement\)](#) ⇒ Element
- [.insertBefore\(srcElement, targetElement\)](#) ⇒ Element
- [.move\(element, offset\)](#) ⇒ Element
- [.moveCentroid\(element, position\)](#) ⇒ Element
- [.rotate\(element, angle\)](#) ⇒ Element
- [.resize\(element, offset\)](#) ⇒ Element
- [.resizeBox\(element, size\)](#) ⇒ Element
- [.clone\(element\)](#) ⇒ Element
- [.getBoundary\(element\)](#) ⇒ [Envelope](#)
- [.getElementById\(id\)](#) ⇒ Element
- [.getElementsByType\(shapeType, excludeType\)](#) ⇒ Array.<Element>
- [.getElementsByShapeId\(shapeId\)](#) ⇒ Array.<Element>
- [.getRelatedElementsFromEdge\(edgeElement\)](#) ⇒ Object
- [.getParent\(Element\)](#) ⇒ Element

- [.getchilds\(element\)](#) ⇒ Array
  - [.getAllShapes\(\)](#) ⇒ Array
  - [.getAllEdges\(\)](#) ⇒ Array
  - [.getBBox\(element\)](#) ⇒ Object
  - [.getRootBBox\(\)](#) ⇒ Object
  - [.getRealRootBBox\(\)](#) ⇒ Object
  - [.isSVG\(\)](#) ⇒ Boolean
  - [.isVML\(\)](#) ⇒ Boolean
  - [.setCustomData\(shapeElement, data\)](#)
  - [.getCustomData\(shapeElement\)](#) ⇒ Object
  - [.setExtCustomData\(shapeElement, data\)](#)
  - [.getExtCustomData\(shapeElement\)](#) ⇒ Object
  - [.toXML\(\)](#) ⇒ String
  - [.toJSON\(\)](#) ⇒ Object
  - [.loadXML\(xml\)](#) ⇒ Object
  - [.loadJSON\(json\)](#) ⇒ Object
  - [.undo\(\)](#)
  - [.redo\(\)](#)
  - [.getPrevEdges\(element\)](#) ⇒ Array.<Element>
  - [.getNextEdges\(element\)](#) ⇒ Array.<Element>
  - [.getPrevShapes\(element\)](#) ⇒ Array.<Element>
  - [.getPrevShapeIds\(element\)](#) ⇒ Array.<String>
  - [.getNextShapes\(element\)](#) ⇒ Array.<Element>
  - [.getNextShapeIds\(element\)](#) ⇒ Array.<String>
  - [.onDrawShape\(callbackFunc\)](#)
  - [.onUndo\(callbackFunc\)](#)
  - [.onRedo\(callbackFunc\)](#)
  - [.onDivideLane\(callbackFunc\)](#)
  - [.onDrawLabel\(callbackFunc\)](#)
  - [.onLabelChanged\(callbackFunc\)](#)
  - [.onBeforeLabelChange\(callbackFunc\)](#)
  - [.onRedrawShape\(callbackFunc\)](#)
  - [.onRemoveShape\(callbackFunc\)](#)
  - [.onRotateShape\(callbackFunc\)](#)
  - [.onMoveShape\(callbackFunc\)](#)
  - [.onResizeShape\(callbackFunc\)](#)
  - [.onBeforeConnectShape\(callbackFunc\)](#)
  - [.onBeforeRemoveShape\(callbackFunc\)](#)
  - [.onConnectShape\(callbackFunc\)](#)
  - [.onDisconnectShape\(callbackFunc\)](#)
  - [.onGroup\(callbackFunc\)](#)
  - [.onUnGroup\(callbackFunc\)](#)
  - [.onCollapsed\(callbackFunc\)](#)
  - [.onExpanded\(callbackFunc\)](#)
- [.handler](#) : object
    - [.EventHandler](#)
      - [new OG.handler.EventHandler\(renderer, config\)](#)
      - [.enableEditLabel\(element\)](#)

- [.setMovable\(element, isMovable\)](#)
- [.setConnectable\(element, guide, isConnectable\)](#)
- [.setResizable\(element, guide, isResizable\)](#)
- [.setClickSelectable\(element, isSelectable\)](#)
- [.setGroupDropable\(element\)](#)
- [.setDragSelectable\(isSelectable\)](#)
- [.setEnableHotKey\(isEnableHotKey\)](#)
- [.enableRootContextMenu\(\)](#)
- [.enableShapeContextMenu\(\)](#)
- [.selectShape\(element\)](#)
- [.selectShapes\(element\)](#)
- [.bringToFront\(\)](#)
- [.sendToBack\(\)](#)
- [.bringForward\(\)](#)
- [.sendBackward\(\)](#)
- [.deleteSelectedShape\(\)](#)
- [.changeShape\(\)](#)
- [.showProperty\(\)](#)
- [.selectAll\(\)](#)
- [.copySelectedShape\(\)](#)
- [.cutSelectedShape\(\)](#)
- [.pasteSelectedShape\(\)](#)
- [.duplicateSelectedShape\(\)](#)
- [.groupSelectedShape\(\)](#)
- [.ungroupSelectedShape\(\)](#)
- [.rotateSelectedShape\(angle\)](#)
- [.setLineWidthSelectedShape\(lineWidth\)](#)
- [.setLineColorSelectedShape\(lineColor\)](#)
- [.setLoopTypeSelectedShape\(lineType\)](#)
- [.setLineStyleSelectedShape\(lineStyle\)](#)
- [.setArrowStartSelectedShape\(arrowType\)](#)
- [.setArrowEndSelectedShape\(arrowType\)](#)
- [.setFillColorSelectedShape\(fillColor\)](#)
- [.setFontFamilySelectedShape\(fontFamily\)](#)
- [.setFontSizeSelectedShape\(fontSize\)](#)
- [.setFontColorSelectedShape\(fontColor\)](#)
- [.setFontSizeSelectedShape\(fontWeight\)](#)
- [.setFontStyleSelectedShape\(fontStyle\)](#)
- [.setTextDecorationSelectedShape\(textDecoration\)](#)
- [.setLabelDirectionSelectedShape\(labelDirection\)](#)
- [.setLabelAngleSelectedShape\(labelAngle\)](#)
- [.setLabelPositionSelectedShape\(labelPosition\)](#)
- [.setLabelVerticalSelectedShape\(verticalAlign\)](#)
- [.setLabelHorizontalSelectedShape\(horizontalAlign\)](#)
- [.setLabelSelectedShape\(label\)](#)
- [.setEdgeFromLabelSelectedShape\(label\)](#)
- [.setEdgeToLabelSelectedShape\(label\)](#)
- [.zoomIn\(\)](#)
- [.zoomOut\(\)](#)
- [.fitWindow\(\)](#)
- [.setConnectGuide\(element, isConnectable\)](#)

- [.layout](#) : object
- [.renderer](#) : object

- [.IRenderer](#)

- [new OG.renderer.IRenderer\(container, containerSize, backgroundColor, backgroundImage, config\)](#)
- [.drawShape\(position, shape, size, style, id\) ⇒ Element](#)
- [.drawGeom\(geometry, style\) ⇒ Element](#)
- [.drawText\(position, text, size, style, id\) ⇒ Element](#)
- [.drawImage\(position, imgSrc, size, style, id\) ⇒ Element](#)
- [.drawEdge\(line, style, id, isSelf\) ⇒ Element](#)
- [.drawLabel\(shapeElement, text, style\) ⇒ Element](#)
- [.drawEdgeLabel\(shapeElement, text, type\) ⇒ Element](#)
- [.redrawShape\(element, excludeEdgeld\)](#)
- [.redrawConnectedEdge\(element\)](#)
- [.connect\(fromTerminal, toTerminal, edge, style, label, preventTrigger\) ⇒ Element](#)
- [.disconnect\(element\)](#)
- [.drawDropOverGuide\(element\)](#)
- [.drawGuide\(element\) ⇒ Object](#)
- [.removeGuide\(element\)](#)
- [.removeAllGuide\(\)](#)
- [.removeConnectGuide\(element\)](#)
- [.removeAllConnectGuide\(\)](#)
- [.removeOtherConnectGuide\(element\)](#)
- [.removeAllVirtualEdge\(\)](#)
- [.drawEdgeGuide\(element\) ⇒ Object](#)
- [.drawRubberBand\(position, size, style\) ⇒ Element](#)
- [.removeRubberBand\(root\)](#)
- [.drawDraggableGuide\(element\) ⇒ Element](#)
- [.drawCollapseGuide\(element\) ⇒ Element](#)
- [.removeCollapseGuide\(element\)](#)
- [.group\(elements\) ⇒ Element](#)
- [.ungroup\(groupElements\) ⇒ Array.<Element>](#)
- [.addToGroup\(groupElement, elements\)](#)
- [.collapse\(element\)](#)
- [.expand\(element\)](#)
- [.clear\(\)](#)
- [.removeShape\(element\)](#)
- [.remove\(element\)](#)
- [.removeChild\(element\)](#)
- [.getRootElement\(\) ⇒ Element](#)
- [.getRootGroup\(\) ⇒ Element](#)
- [.getElementByPoint\(position\) ⇒ Element](#)
- [.getElementsByBBox\(envelope\) ⇒ Array.<Element>](#)
- [.setAttr\(element, attribute\)](#)
- [.getAttr\(element, attrName\) ⇒ Object](#)
- [.setShapeStyle\(element, style\)](#)
- [.toFront\(element\)](#)
- [.toBack\(element\)](#)

- [.bringForward\(element\)](#)
- [.sendBackward\(element\)](#)
- [.getCanvasSize\(\)](#) ⇒ Array.<Number>
- [.setCanvasSize\(size\)](#)
- [.fitCanvasSize\(minSize, fitScale\)](#)
- [.setViewBox\(position, size, isFit\)](#)
- [.getScale\(\)](#) ⇒ Number
- [.setScale\(scale\)](#)
- [.show\(element\)](#)
- [.hide\(element\)](#)
- [.appendChild\(srcElement, targetElement\)](#) ⇒ Element
- [.insertAfter\(srcElement, targetElement\)](#) ⇒ Element
- [.insertBefore\(srcElement, targetElement\)](#) ⇒ Element
- [.move\(element, offset\)](#) ⇒ Element
- [.moveCentroid\(element, position\)](#) ⇒ Element
- [.rotate\(element, angle\)](#) ⇒ Element
- [.resize\(element, offset\)](#) ⇒ Element
- [.resizeBox\(element, size\)](#) ⇒ Element
- [.clone\(element\)](#) ⇒ Element
- [.getElementById\(id\)](#) ⇒ Element
- [.getElementsByType\(shapeType, excludeType\)](#) ⇒ Array.<Element>
- [.getBBox\(element\)](#) ⇒ Object
- [.getRootBBox\(\)](#) ⇒ Object
- [.getRealRootBBox\(\)](#) ⇒ Object
- [.getContainer\(\)](#) ⇒ Element
- [.isSVG\(\)](#) ⇒ Boolean
- [.isVML\(\)](#) ⇒ Boolean
- [.getPrevEdges\(element\)](#) ⇒ Array.<Element>
- [.getNextEdges\(element\)](#) ⇒ Array.<Element>
- [.getPrevShapes\(element\)](#) ⇒ Array.<Element>
- [.getPrevShapeIds\(element\)](#) ⇒ Array.<String>
- [.getNextShapes\(element\)](#) ⇒ Array.<Element>
- [.getNextShapeIds\(element\)](#) ⇒ Array.<String>
- [.getConnectGuideElements\(Element\)](#) ⇒ Array
- [.isTopGroup\(Element\)](#) ⇒ boolean
- [.getParent\(element\)](#) ⇒ Element
- [.getChilds\(element\)](#) ⇒ Array
- [.isGroup\(element\)](#) ⇒ boolean
- [.getAllShapes\(\)](#) ⇒ Array
- [.getAllEdges\(\)](#) ⇒ Array
- [.getAllNotEdges\(\)](#) ⇒ Array
- [.isEdge\(\)](#) ⇒ boolean
- [.isShape\(\)](#) ⇒ boolean
- [.initHistory\(\)](#)
- [.addHistory\(\)](#)
- [.undo\(\)](#)
- [.redo\(\)](#)

- [.RaphaelRenderer](#) ⇐ [IRenderer](#)

- [new OG.renderer.RaphaelRenderer\(container, containerSize, backgroundColor, backgroundImage, config\)](#)
- [.drawHtml\(position, html, size, style, id\)](#) ⇒ Element
- [.getPointOfInflectionFromEdge\(\)](#)
- [.reconnect\(edge\)](#) ⇒ Element
- [.disconnectOneWay\(element, connectDirection\)](#)
- [.drawStickGuide\(element, position\)](#)
- [.setTextListInController\(element, textList\)](#)
- [.getTextListInController\(element\)](#)
- [.getConnectGuideElements\(Element\)](#) ⇒ Array
- [.getNotConnectGuideElements\(Element\)](#) ⇒ Array
- [.removeConnectGuide\(element\)](#)
- [.removeAllConnectGuide\(\)](#)
- [.removeOtherConnectGuide\(element\)](#)
- [.getSpots\(element\)](#) ⇒ Array
- [.getCircleSpots\(element\)](#) ⇒ Array
- [.createVirtualSpot\(x, x, element\)](#) ⇒ Element
- [.getVirtualSpot\(element\)](#) ⇒ Element
- [.removeVirtualSpot\(element\)](#) ⇒ Element
- [.selectSpot\(선택한\)](#)
- [.getChildNodes\(element\)](#) ⇒ Array
- [.trimEdge\(element\)](#)
- [.trimConnectInnerVertice\(element\)](#) ⇒ Element
- [.trimConnectIntersection\(element\)](#) ⇒ Element
- [.getBoundary\(element\)](#) ⇒ Envelope
- [.setHighlight\(element, highlight\)](#)
- [.removeHighlight\(element, highlight\)](#)
- [.createTerminalString\(Element, point\)](#) ⇒ String
- [.createDefaultTerminalString\(Element\)](#) ⇒ String
- [.toFrontEdges\(\)](#)
- [.removeAllEdgeGuide\(\)](#)
- [.createVirtualEdge\(x, x, targetEle\)](#) ⇒ Element
- [.updateVirtualEdge\(x, x\)](#)
- [.getTargetfromVirtualEdge\(x, x\)](#)
- [.removeAllVirtualEdge\(\)](#)
- [.isLane\(Element\)](#) ⇒ boolean
- [.isPool\(Element\)](#) ⇒ boolean
- [.isScopeActivity\(Element\)](#) ⇒ boolean
- [.isHorizontalLane\(Element\)](#) ⇒ boolean
- [.isVerticalLane\(Element\)](#) ⇒ boolean
- [.isHorizontalPool\(Element\)](#) ⇒ boolean
- [.isVerticalPool\(Element\)](#) ⇒ boolean
- [.getChildLane\(Element\)](#) ⇒ Array
- [.enableDivideCount\(Element\)](#) ⇒ Number
- [.getExceptTitleLaneArea\(Element, boundary\)](#)
- [.divideLane\(Element, quarterOrder\)](#)
- [.getBaseLanes\(Element\)](#) ⇒ Array
- [.getRootLane\(Element\)](#) ⇒ Element

- [.getIndexOfLane\(Element\) ⇒ Number](#)
- [.getDepthOfLane\(Element\) ⇒ Number](#)
- [.reEstablishLane\(Element\)](#)
- [.getBoundaryOfElements\(elements\) ⇒ Envelope](#)
- [.getNearestBaseLaneIndexAsDirection\(Element, direction\) ⇒ Number](#)
- [.getBoundaryOfInnerShapesGroup\(Element\) ⇒ Envelope](#)
- [.getSmallestBaseLane\(Element, baseLane\)](#)
- [.resizeLane\(Element, offset\)](#)
- [.removeLaneShape\(Element\)](#)
- [.getInnerShapesOfLane\(Element\)](#)
- [.fitLaneOrder\(Element\)](#)
- [.getRootGroupOfShape\(Element\) ⇒ Element](#)
- [.checkBridgeEdge\(Element\)](#)
- [.checkAllBridgeEdge\(\)](#)
- [.getInnerShapesOfGroup\(Element\)](#)
- [.getFrontForCoordinate\(point\) ⇒ Element](#)
- [.getFrontForBoundary\(boundary\) ⇒ Element](#)
- [.trimEdgeDirection\(Edge, FromShape, ToShape\) ⇒ Element](#)
- [.putInnerShapeToPool\(Element\) ⇒ Element](#)
- [.setDropablePool\(Element\) ⇒ Element](#)
- [.offDropablePool\(\)](#)
- [.drawShape\(position, shape, size, style, id\) ⇒ Element](#)
- [.drawGeom\(geometry, style\) ⇒ Element](#)
- [.drawText\(position, text, size, style, id\) ⇒ Element](#)
- [.drawImage\(position, imgSrc, size, style, id\) ⇒ Element](#)
- [.drawEdge\(line, style, id, isSelf\) ⇒ Element](#)
- [.drawLabel\(shapeElement, text, style\) ⇒ Element](#)
  - [~getCenterOfEdge\(element\) ⇒ OG.Coordinate](#)
- [.drawEdgeLabel\(shapeElement, text, type\) ⇒ Element](#)
- [.redrawShape\(element, excludeEdgeld\)](#)
- [.redrawConnectedEdge\(element\)](#)
- [.connect\(fromTerminal, toTerminal, edge, style, label, preventTrigger\) ⇒ Element](#)
- [.disconnect\(element\)](#)
- [.drawDropOverGuide\(element\)](#)
- [.drawGuide\(element\) ⇒ Object](#)
- [.removeGuide\(element\)](#)
- [.removeAllGuide\(\)](#)
- [.drawEdgeGuide\(element\) ⇒ Object](#)
- [.drawRubberBand\(position, size, style\) ⇒ Element](#)
- [.removeRubberBand\(root\)](#)
- [.drawDraggableGuide\(element\) ⇒ Element](#)
- [.drawCollapseGuide\(element\) ⇒ Element](#)
- [.removeCollapseGuide\(element\)](#)
- [.group\(elements\) ⇒ Element](#)
- [.ungroup\(groupElements\) ⇒ Array.<Element>](#)
- [.addToGroup\(groupElement, elements\)](#)

- [.collapse\(element\)](#)
- [.expand\(element\)](#)
- [.clear\(\)](#)
- [.removeShape\(element\)](#)
- [.remove\(element\)](#)
- [.removeChild\(element\)](#)
- [.getRootElement\(\) ⇒ Element](#)
- [.getRootElement\(\) ⇒ Element](#)
- [.getElementByPoint\(position\) ⇒ Element](#)
- [.getElementsByBBox\(envelope\) ⇒ Array.<Element>](#)
- [.setAttr\(element, attribute\)](#)
- [.getAttr\(element, attrName\) ⇒ Object](#)
- [.setShapeStyle\(element, style\)](#)
- [.toFront\(element\)](#)
- [.toBack\(element\)](#)
- [.bringForward\(element\)](#)
- [.sendBackward\(element\)](#)
- [.getCanvasSize\(\) ⇒ Array.<Number>](#)
- [.setCanvasSize\(size\)](#)
- [.fitCanvasSize\(minSize, fitScale\)](#)
- [.setViewBox\(position, size, isFit\)](#)
- [.getScale\(\) ⇒ Number](#)
- [.setScale\(scale\)](#)
- [.show\(element\)](#)
- [.hide\(element\)](#)
- [.appendChild\(srcElement, targetElement\) ⇒ Element](#)
- [.insertAfter\(srcElement, targetElement\) ⇒ Element](#)
- [.insertBefore\(srcElement, targetElement\) ⇒ Element](#)
- [.move\(element, offset\) ⇒ Element](#)
- [.moveCentroid\(element, position\) ⇒ Element](#)
- [.rotate\(element, angle\) ⇒ Element](#)
- [.resize\(element, offset\) ⇒ Element](#)
- [.resizeBox\(element, size\) ⇒ Element](#)
- [.clone\(element\) ⇒ Element](#)
- [.getElementById\(id\) ⇒ Element](#)
- [.getElementsByType\(shapeType, excludeType\) ⇒ Array.<Element>](#)
- [.getBBox\(element\) ⇒ Object](#)
- [.getRootBBox\(\) ⇒ Object](#)
- [.getRealRootBBox\(\) ⇒ Object](#)
- [.getContainer\(\) ⇒ Element](#)
- [.isSVG\(\) ⇒ Boolean](#)
- [.isVML\(\) ⇒ Boolean](#)
- [.getPrevEdges\(element\) ⇒ Array.<Element>](#)
- [.getNextEdges\(element\) ⇒ Array.<Element>](#)
- [.getPrevShapes\(element\) ⇒ Array.<Element>](#)
- [.getPrevShapeIds\(element\) ⇒ Array.<String>](#)
- [.getNextShapes\(element\) ⇒ Array.<Element>](#)
- [.getNextShapeIds\(element\) ⇒ Array.<String>](#)

- [.isTopGroup\(Element\)](#) ⇒ boolean
  - [.getParent\(element\)](#) ⇒ Element
  - [.getChilds\(element\)](#) ⇒ Array
  - [.isGroup\(element\)](#) ⇒ boolean
  - [.getAllShapes\(\)](#) ⇒ Array
  - [.getAllEdges\(\)](#) ⇒ Array
  - [.getAllNotEdges\(\)](#) ⇒ Array
  - [.isEdge\(\)](#) ⇒ boolean
  - [.isShape\(\)](#) ⇒ boolean
  - [.initHistory\(\)](#)
  - [.addHistory\(\)](#)
  - [.undo\(\)](#)
  - [.redo\(\)](#)
- [.marker](#) : object
    - [.ArrowMarker](#) ⇐ [IMarker](#)
      - [new OG.marker.ArrowMarker\(\)](#)
      - [.MARKER\\_ID](#) : String
      - [.geom](#) : [Geometry](#)
      - [.createMarker\(\)](#) ⇒ \*
    - [.CircleMarker](#) ⇐ [IMarker](#)
      - [new OG.marker.CircleMarker\(\)](#)
      - [.MARKER\\_ID](#) : String
      - [.geom](#) : [Geometry](#)
      - [.createMarker\(\)](#) ⇒ \*
    - [.IMarker](#)
      - [new OG.marker.IMarker\(\)](#)
      - [.MARKER\\_ID](#) : String
      - [.geom](#) : [Geometry](#)
      - [.createMarker\(\)](#) ⇒ \*
    - [.RectangleMarker](#) ⇐ [IMarker](#)
      - [new OG.marker.RectangleMarker\(\)](#)
      - [.MARKER\\_ID](#) : String
      - [.geom](#) : [Geometry](#)
      - [.createMarker\(\)](#) ⇒ \*
    - [.SwitchLMarker](#) ⇐ [IMarker](#)
      - [new OG.marker.SwitchLMarker\(\)](#)
      - [.MARKER\\_ID](#) : String
      - [.geom](#) : [Geometry](#)
      - [.createMarker\(\)](#) ⇒ \*
    - [.SwitchRMarker](#) ⇐ [IMarker](#)

- [new OG.marker.SwitchRMarker\(\)](#)
  - [.MARKER\\_ID](#) : String
  - [.geom](#) : Geometry
  - [.createMarker\(\)](#) ⇒ \*
- [.SwitchXMarker](#) ⇐ IMarker
  - [new OG.marker.SwitchXMarker\(\)](#)
  - [.MARKER\\_ID](#) : String
  - [.geom](#) : Geometry
  - [.createMarker\(\)](#) ⇒ \*
- [.pattern](#) : object
  - [.HatchedPattern](#) ⇐ IPattern
    - [new OG.pattern.HatchedPattern\(\)](#)
    - [.PATTERN\\_ID](#) : String
    - [.geom](#) : Geometry
    - [.createPattern\(\)](#) ⇒ \*
  - [.IPattern](#)
    - [new OG.pattern.IPattern\(\)](#)
    - [.PATTERN\\_ID](#) : String
    - [.geom](#) : Geometry
    - [.createPattern\(\)](#) ⇒ \*
  - [.RectPattern](#) ⇐ IPattern
    - [new OG.pattern.RectPattern\(\)](#)
    - [.PATTERN\\_ID](#) : String
    - [.geom](#) : Geometry
    - [.createPattern\(\)](#) ⇒ \*
- [.shape](#) : object
  - [.CircleShape](#) ⇐ GeomShape
    - [new OG.shape.CircleShape\(label\)](#)
    - [.TYPE](#) : String
    - [.SHAPE\\_ID](#) : String
    - [.geom](#) : Geometry
    - [.label](#) : String
    - [.isCollapsed](#) : Boolean
    - [.SELECTABLE](#) : Boolean
    - [.MOVABLE](#) : Boolean
    - [.RESIZABLE](#) : Boolean
    - [.CONNECTABLE](#) : Boolean
    - [.ENABLE\\_FROM](#) : Boolean
    - [.ENABLE\\_TO](#) : Boolean
    - [.SELF\\_CONNECTABLE](#) : Boolean
    - [.CONNECT\\_CLONEABLE](#) : Boolean

- .CONNECT\_REQUIRED : Boolean
- .CONNECT\_STYLE\_CHANGE : Boolean
- .DELETABLE : Boolean
- .LABEL\_EDITABLE : Boolean
- .data : Object
- .textList : Array
- .clone() ⇒ IShape
- .createShape() ⇒ \*
  
- .EdgeShape ⇐ IShape
  - new OG.shape.EdgeShape(from, to, label, fromLabel, toLabel)
  - .from : Array.<Number>
  - .to : Array.<Number>
  - .fromLabel : String
  - .toLabel : String
  - .TYPE : String
  - .SHAPE\_ID : String
  - .geom : Geometry
  - .label : String
  - .isCollapsed : Boolean
  - .SELECTABLE : Boolean
  - .MOVABLE : Boolean
  - .RESIZABLE : Boolean
  - .CONNECTABLE : Boolean
  - .ENABLE\_FROM : Boolean
  - .ENABLE\_TO : Boolean
  - .SELF\_CONNECTABLE : Boolean
  - .CONNECT\_CLONEABLE : Boolean
  - .CONNECT\_REQUIRED : Boolean
  - .CONNECT\_STYLE\_CHANGE : Boolean
  - .DELETABLE : Boolean
  - .LABEL\_EDITABLE : Boolean
  - .data : Object
  - .textList : Array
  - .createShape() ⇒ \*
  - .clone() ⇒ IShape
  
- .EllipseShape ⇐ GeomShape
  - new OG.shape.EllipseShape(label)
  - .TYPE : String
  - .SHAPE\_ID : String
  - .geom : Geometry
  - .label : String
  - .isCollapsed : Boolean
  - .SELECTABLE : Boolean
  - .MOVABLE : Boolean
  - .RESIZABLE : Boolean
  - .CONNECTABLE : Boolean

- [.ENABLE\\_FROM](#) : Boolean
- [.ENABLE\\_TO](#) : Boolean
- [.SELF\\_CONNECTABLE](#) : Boolean
- [.CONNECT\\_CLONEABLE](#) : Boolean
- [.CONNECT\\_REQUIRED](#) : Boolean
- [.CONNECT\\_STYLE\\_CHANGE](#) : Boolean
- [.DELETABLE](#) : Boolean
- [.LABEL\\_EDITABLE](#) : Boolean
- [.data](#) : Object
- [.textList](#) : Array
- [.clone\(\)](#) ⇒ IShape
- [.createShape\(\)](#) ⇒ \*
  
- [.GeomShape](#) ⇐ IShape
  - [new OG.shape.GemShape\(\)](#)
  - [.TYPE](#) : String
  - [.SHAPE\\_ID](#) : String
  - [.geom](#) : Geometry
  - [.label](#) : String
  - [.isCollapsed](#) : Boolean
  - [.SELECTABLE](#) : Boolean
  - [.MOVABLE](#) : Boolean
  - [.RESIZABLE](#) : Boolean
  - [.CONNECTABLE](#) : Boolean
  - [.ENABLE\\_FROM](#) : Boolean
  - [.ENABLE\\_TO](#) : Boolean
  - [.SELF\\_CONNECTABLE](#) : Boolean
  - [.CONNECT\\_CLONEABLE](#) : Boolean
  - [.CONNECT\\_REQUIRED](#) : Boolean
  - [.CONNECT\\_STYLE\\_CHANGE](#) : Boolean
  - [.DELETABLE](#) : Boolean
  - [.LABEL\\_EDITABLE](#) : Boolean
  - [.data](#) : Object
  - [.textList](#) : Array
  - [.createShape\(\)](#) ⇒ \*
  - [.clone\(\)](#) ⇒ IShape
  
- [.GroupShape](#) ⇐ IShape
  - [new OG.shape.GroupShape\(label\)](#)
  - [.GROUP\\_DROPABLE](#) : Boolean
  - [.GROUP\\_COLLAPSIBLE](#) : Boolean
  - [.TYPE](#) : String
  - [.SHAPE\\_ID](#) : String
  - [.geom](#) : Geometry
  - [.label](#) : String
  - [.isCollapsed](#) : Boolean
  - [.SELECTABLE](#) : Boolean
  - [.MOVABLE](#) : Boolean

- [.RESIZABLE](#) : Boolean
- [.CONNECTABLE](#) : Boolean
- [.ENABLE FROM](#) : Boolean
- [.ENABLE TO](#) : Boolean
- [.SELF CONNECTABLE](#) : Boolean
- [.CONNECT CLONEABLE](#) : Boolean
- [.CONNECT REQUIRED](#) : Boolean
- [.CONNECT STYLE CHANGE](#) : Boolean
- [.DELETABLE](#) : Boolean
- [.LABEL EDITABLE](#) : Boolean
- [.data](#) : Object
- [.textList](#) : Array
- [.createShape\(\)](#) => \*
- [.clone\(\)](#) => IShape

▪ [.HorizontalLaneShape](#) ⇐ GroupShape

- [new OG.shape.HorizontalLaneShape\(label\)](#)
- [.GROUP DROPABLE](#) : Boolean
- [.GROUP COLLAPSIBLE](#) : Boolean
- [.TYPE](#) : String
- [.SHAPE ID](#) : String
- [.geom](#) : Geometry
- [.label](#) : String
- [.isCollapsed](#) : Boolean
- [.SELECTABLE](#) : Boolean
- [.MOVABLE](#) : Boolean
- [.RESIZABLE](#) : Boolean
- [.CONNECTABLE](#) : Boolean
- [.ENABLE FROM](#) : Boolean
- [.ENABLE TO](#) : Boolean
- [.SELF CONNECTABLE](#) : Boolean
- [.CONNECT CLONEABLE](#) : Boolean
- [.CONNECT REQUIRED](#) : Boolean
- [.CONNECT STYLE CHANGE](#) : Boolean
- [.DELETABLE](#) : Boolean
- [.LABEL EDITABLE](#) : Boolean
- [.data](#) : Object
- [.textList](#) : Array
- [.createShape\(\)](#) => \*
- [.clone\(\)](#) => IShape

▪ [.HorizontalPoolShape](#) ⇐ GroupShape

- [new OG.shape.HorizontalPoolShape\(label\)](#)
- [.GROUP DROPABLE](#) : Boolean
- [.GROUP COLLAPSIBLE](#) : Boolean
- [.TYPE](#) : String
- [.SHAPE ID](#) : String
- [.geom](#) : Geometry

- [.label](#) : String
- [.isCollapsed](#) : Boolean
- [.SELECTABLE](#) : Boolean
- [.MOVABLE](#) : Boolean
- [.RESIZABLE](#) : Boolean
- [.CONNECTABLE](#) : Boolean
- [.ENABLE FROM](#) : Boolean
- [.ENABLE TO](#) : Boolean
- [.SELF CONNECTABLE](#) : Boolean
- [.CONNECT CLONEABLE](#) : Boolean
- [.CONNECT REQUIRED](#) : Boolean
- [.CONNECT STYLE CHANGE](#) : Boolean
- [.DELETABLE](#) : Boolean
- [.LABEL EDITABLE](#) : Boolean
- [.data](#) : Object
- [.textList](#) : Array
- [.createShape\(\)](#) ⇒ \*
- [.clone\(\)](#) ⇒ IShape
  
- [.HtmlShape](#) ⇐ IShape
  - [new OG.shape.HtmlShape\(html, label\)](#)
  - [.html](#) : String
  - [.angle](#) : Number
  - [.TYPE](#) : String
  - [.SHAPE\\_ID](#) : String
  - [.geom](#) : Geometry
  - [.label](#) : String
  - [.isCollapsed](#) : Boolean
  - [.SELECTABLE](#) : Boolean
  - [.MOVABLE](#) : Boolean
  - [.RESIZABLE](#) : Boolean
  - [.CONNECTABLE](#) : Boolean
  - [.ENABLE FROM](#) : Boolean
  - [.ENABLE TO](#) : Boolean
  - [.SELF CONNECTABLE](#) : Boolean
  - [.CONNECT CLONEABLE](#) : Boolean
  - [.CONNECT REQUIRED](#) : Boolean
  - [.CONNECT STYLE CHANGE](#) : Boolean
  - [.DELETABLE](#) : Boolean
  - [.LABEL EDITABLE](#) : Boolean
  - [.data](#) : Object
  - [.textList](#) : Array
  - [.createShape\(\)](#) ⇒ \*
  - [.clone\(\)](#) ⇒ IShape
  
- [.IShape](#)
  - [new OG.shape.IShape\(\)](#)
  - [.TYPE](#) : String

- [.SHAPE\\_ID](#) : String
  - [.geom](#) : [Geometry](#)
  - [.label](#) : String
  - [.isCollapsed](#) : Boolean
  - [.SELECTABLE](#) : Boolean
  - [.MOVABLE](#) : Boolean
  - [.RESIZABLE](#) : Boolean
  - [.CONNECTABLE](#) : Boolean
  - [.ENABLE FROM](#) : Boolean
  - [.ENABLE TO](#) : Boolean
  - [.SELF\\_CONNECTABLE](#) : Boolean
  - [.CONNECT\\_CLONEABLE](#) : Boolean
  - [.CONNECT\\_REQUIRED](#) : Boolean
  - [.CONNECT\\_STYLE\\_CHANGE](#) : Boolean
  - [.DELETABLE](#) : Boolean
  - [.LABEL\\_EDITABLE](#) : Boolean
  - [.data](#) : Object
  - [.textList](#) : Array
  - [.createShape\(\)](#) ⇒ \*
  - [.clone\(\)](#) ⇒ [IShape](#)
- [.ImageShape](#) ⇐ [IShape](#)
- [new OG.shape.ImageShape\(image, label\)](#)
  - [.image](#) : String
  - [.angle](#) : Number
  - [.TYPE](#) : String
  - [.SHAPE\\_ID](#) : String
  - [.geom](#) : [Geometry](#)
  - [.label](#) : String
  - [.isCollapsed](#) : Boolean
  - [.SELECTABLE](#) : Boolean
  - [.MOVABLE](#) : Boolean
  - [.RESIZABLE](#) : Boolean
  - [.CONNECTABLE](#) : Boolean
  - [.ENABLE FROM](#) : Boolean
  - [.ENABLE TO](#) : Boolean
  - [.SELF\\_CONNECTABLE](#) : Boolean
  - [.CONNECT\\_CLONEABLE](#) : Boolean
  - [.CONNECT\\_REQUIRED](#) : Boolean
  - [.CONNECT\\_STYLE\\_CHANGE](#) : Boolean
  - [.DELETABLE](#) : Boolean
  - [.LABEL\\_EDITABLE](#) : Boolean
  - [.data](#) : Object
  - [.textList](#) : Array
  - [.createShape\(\)](#) ⇒ \*
  - [.clone\(\)](#) ⇒ [IShape](#)
- [.RectangleShape](#) ⇐ [GeomShape](#)

- [new OG.shape.RectangleShape\(label\)](#)
- [.TYPE : String](#)
- [.SHAPE\\_ID : String](#)
- [.geom : Geometry](#)
- [.label : String](#)
- [.isCollapsed : Boolean](#)
- [.SELECTABLE : Boolean](#)
- [.MOVABLE : Boolean](#)
- [.RESIZABLE : Boolean](#)
- [.CONNECTABLE : Boolean](#)
- [.ENABLE FROM : Boolean](#)
- [.ENABLE TO : Boolean](#)
- [.SELF CONNECTABLE : Boolean](#)
- [.CONNECT CLONEABLE : Boolean](#)
- [.CONNECT REQUIRED : Boolean](#)
- [.CONNECT STYLE CHANGE : Boolean](#)
- [.DELETABLE : Boolean](#)
- [.LABEL EDITABLE : Boolean](#)
- [.data : Object](#)
- [.textList : Array](#)
- [.clone\(\) => IShape](#)
- [.createShape\(\) => \\*](#)
  
- [.SpotShape ⇐ GeomShape](#)
  - [new OG.shape.SpotShape\(label\)](#)
  - [.TYPE : String](#)
  - [.SHAPE\\_ID : String](#)
  - [.geom : Geometry](#)
  - [.label : String](#)
  - [.isCollapsed : Boolean](#)
  - [.SELECTABLE : Boolean](#)
  - [.MOVABLE : Boolean](#)
  - [.RESIZABLE : Boolean](#)
  - [.CONNECTABLE : Boolean](#)
  - [.ENABLE FROM : Boolean](#)
  - [.ENABLE TO : Boolean](#)
  - [.SELF CONNECTABLE : Boolean](#)
  - [.CONNECT CLONEABLE : Boolean](#)
  - [.CONNECT REQUIRED : Boolean](#)
  - [.CONNECT STYLE CHANGE : Boolean](#)
  - [.DELETABLE : Boolean](#)
  - [.LABEL EDITABLE : Boolean](#)
  - [.data : Object](#)
  - [.textList : Array](#)
  - [.clone\(\) => IShape](#)
  - [.createShape\(\) => \\*](#)
  
- [.TextShape ⇐ IShape](#)

- [new OG.shape.TextShape\(text\)](#)
- [.text](#) : String
- [.angle](#) : Number
- [.TYPE](#) : String
- [.SHAPE\\_ID](#) : String
- [.geom](#) : [Geometry](#)
- [.label](#) : String
- [.isCollapsed](#) : Boolean
- [.SELECTABLE](#) : Boolean
- [.MOVABLE](#) : Boolean
- [.RESIZABLE](#) : Boolean
- [.CONNECTABLE](#) : Boolean
- [.ENABLE FROM](#) : Boolean
- [.ENABLE TO](#) : Boolean
- [.SELF CONNECTABLE](#) : Boolean
- [.CONNECT CLONEABLE](#) : Boolean
- [.CONNECT REQUIRED](#) : Boolean
- [.CONNECT STYLE CHANGE](#) : Boolean
- [.DELETABLE](#) : Boolean
- [.LABEL\\_EDITABLE](#) : Boolean
- [.data](#) : Object
- [.textList](#) : Array
- [.createShape\(\)](#) ⇒ \*
- [.clone\(\)](#) ⇒ [IShape](#)
  
- [.VerticalLaneShape](#) ⇐ [GroupShape](#)
  - [new OG.shape.VerticalLaneShape\(label\)](#)
  - [.GROUP\\_DROPABLE](#) : Boolean
  - [.GROUP\\_COLLAPSIBLE](#) : Boolean
  - [.TYPE](#) : String
  - [.SHAPE\\_ID](#) : String
  - [.geom](#) : [Geometry](#)
  - [.label](#) : String
  - [.isCollapsed](#) : Boolean
  - [.SELECTABLE](#) : Boolean
  - [.MOVABLE](#) : Boolean
  - [.RESIZABLE](#) : Boolean
  - [.CONNECTABLE](#) : Boolean
  - [.ENABLE FROM](#) : Boolean
  - [.ENABLE TO](#) : Boolean
  - [.SELF\\_CONNECTABLE](#) : Boolean
  - [.CONNECT\\_CLONEABLE](#) : Boolean
  - [.CONNECT\\_REQUIRED](#) : Boolean
  - [.CONNECT\\_STYLE\\_CHANGE](#) : Boolean
  - [.DELETABLE](#) : Boolean
  - [.LABEL\\_EDITABLE](#) : Boolean
  - [.data](#) : Object
  - [.textList](#) : Array

- [.createShape\(\)](#) ⇒ \*
- [.clone\(\)](#) ⇒ [IShape](#)
- [.VerticalPoolShape](#) ⇐ [GroupShape](#)
  - [new OG.shape.VerticalPoolShape\(label\)](#)
  - [.GROUP\\_DROPABLE](#) : Boolean
  - [.GROUP\\_COLLAPSIBLE](#) : Boolean
  - [.TYPE](#) : String
  - [.SHAPE\\_ID](#) : String
  - [.geom](#) : [Geometry](#)
  - [.label](#) : String
  - [.isCollapsed](#) : Boolean
  - [.SELECTABLE](#) : Boolean
  - [.MOVABLE](#) : Boolean
  - [.RESIZABLE](#) : Boolean
  - [.CONNECTABLE](#) : Boolean
  - [.ENABLE\\_FROM](#) : Boolean
  - [.ENABLE\\_TO](#) : Boolean
  - [.SELF\\_CONNECTABLE](#) : Boolean
  - [.CONNECT\\_CLONEABLE](#) : Boolean
  - [.CONNECT\\_REQUIRED](#) : Boolean
  - [.CONNECT\\_STYLE\\_CHANGE](#) : Boolean
  - [.DELETABLE](#) : Boolean
  - [.LABEL\\_EDITABLE](#) : Boolean
  - [.data](#) : Object
  - [.textList](#) : Array
  - [.createShape\(\)](#) ⇒ \*
  - [.clone\(\)](#) ⇒ [IShape](#)
- [.bpmn](#) : object
- [.elec](#) : object

## OG.common : object

Kind: static namespace of [OG](#)

- [.common](#) : object
  - [.Constants](#)
    - [new OG.common.Constants\(\)](#)
    - [.GEOM\\_TYPE](#)
    - [.GEOM\\_NAME](#)
    - [.NUM\\_PRECISION](#)
    - [.NODE\\_TYPE](#)
    - [.SHAPE\\_TYPE](#)
    - [.EDGE\\_TYPE](#)
    - [.LABEL\\_SUFFIX](#)
    - [.LABEL\\_EDITOR\\_SUFFIX](#)
    - [.FROM\\_LABEL\\_SUFFIX](#)

- [.TO\\_LABEL\\_SUFFIX](#)
- [.RUBBER\\_BAND\\_ID](#)
- [.RUBBER\\_BAND\\_TOLERANCE](#)
- [.GUIDE\\_SUFFIX](#)
- [.COLLAPSE\\_SUFFIX](#)
- [.LOOPTYPE\\_SUFFIX](#)
- [.TASKTYPE\\_SUFFIX](#)
- [.INCLUSION\\_SUFFIX](#)
- [.STATUS\\_SUFFIX](#)
- [.EXCEPTIONTYPE\\_SUFFIX](#)
- [.MOVE\\_SNAP\\_SIZE](#)
- [.DROP\\_OVER\\_BBOX\\_SUFFIX](#)
- [.TERMINAL\\_SUFFIX](#)
- [.TERMINAL](#)
- [.MARKER\\_TEMP\\_NODE](#)
- [.PATTERN\\_TEMP\\_NODE](#)
- [.MARKER\\_DEFS\\_SUFFIX](#)
- [.ORIGINAL\\_NODE](#)
- [.CONNECT\\_GUIDE\\_EVENT\\_AREA](#)
- [.CONNECT\\_GUIDE\\_SUFFIX](#)

- [.CurveUtil](#)

- [new OG.common.CurveUtil\(\)](#)
- [.CatmullRomSpline\(points\) ⇒ Object](#)
- [.Bezier\(points\) ⇒ Object](#)
- [.BSpline\(points, order\) ⇒ Object](#)

- [.HashMap](#)

- [new OG.common.HashMap\(jsonObject\)](#)
- [.map : Object](#)
- [.put\(key, value\)](#)
- [.get\(key\) ⇒ Object](#)
- [.containsKey\(key\) ⇒ Boolean](#)
- [.containsValue\(value\) ⇒ Boolean](#)
- [.isEmpty\(\) ⇒ Boolean](#)
- [.clear\(\)](#)
- [.remove\(key\)](#)
- [.keys\(\) ⇒ Array.<String>](#)
- [.values\(\) ⇒ Array.<Object>](#)
- [.size\(\) ⇒ Number](#)
- [.toString\(\) ⇒ String](#)

- [.JSON](#)

- [new OG.common.JSON\(\)](#)
- [.encode ⇒ String](#)
- [.decode ⇒ Object](#)
- [.encodeDate\(d\) ⇒ String](#)

- [.Util](#)

- [new OG.common.Util\(\)](#)
- [.extend](#) ⇒ Function
- [.clone\(obj\)](#) ⇒ Object
- [.round\(val\)](#) ⇒ Number
- [.roundPrecision\(val, precision\)](#) ⇒ Number
- [.roundGrid\(val, snapSize\)](#) ⇒ Number
- [.apply\(obj, config, defaults\)](#) ⇒ Object

## common.Constants

Kind: static class of [common](#)

Author: [Seunqpil Park \(mailto:sppark@uengine.org\)](mailto:Seunqpil.Park@sppark@uengine.org)

- [Constants](#)

- [new OG.common.Constants\(\)](#)
- [.GEOM\\_TYPE](#)
- [.GEOM\\_NAME](#)
- [.NUM\\_PRECISION](#)
- [.NODE\\_TYPE](#)
- [.SHAPE\\_TYPE](#)
- [.EDGE\\_TYPE](#)
- [.LABEL\\_SUFFIX](#)
- [.LABEL\\_EDITOR\\_SUFFIX](#)
- [.FROM\\_LABEL\\_SUFFIX](#)
- [.TO\\_LABEL\\_SUFFIX](#)
- [.RUBBER\\_BAND\\_ID](#)
- [.RUBBER\\_BAND\\_TOLERANCE](#)
- [.GUIDE\\_SUFFIX](#)
- [.COLLAPSE\\_SUFFIX](#)
- [.LOOPTYPE\\_SUFFIX](#)
- [.TASKTYPE\\_SUFFIX](#)
- [.INCLUSION\\_SUFFIX](#)
- [.STATUS\\_SUFFIX](#)
- [.EXCEPTIONTYPE\\_SUFFIX](#)
- [.MOVE\\_SNAP\\_SIZE](#)
- [.DROP\\_OVER\\_BBOX\\_SUFFIX](#)
- [.TERMINAL\\_SUFFIX](#)
- [.TERMINAL](#)
- [.MARKER\\_TEMP\\_NODE](#)
- [.PATTERN\\_TEMP\\_NODE](#)
- [.MARKER\\_DEFS\\_SUFFIX](#)
- [.ORIGINAL\\_NODE](#)
- [.CONNECT\\_GUIDE\\_EVENT\\_AREA](#)
- [.CONNECT\\_GUIDE\\_SUFFIX](#)

**new OG.common.Constants()**

공통 상수 정의 Javascript 클래스

**Constants.GEOM\_TYPE**

공간 기하 객체 타입 정의

**Kind:** static property of [Constants](#)

**Constants.GEOM\_NAME**

공간 기하 객체 타입-이름 매핑

**Kind:** static property of [Constants](#)

**Constants.NUM\_PRECISION**

숫자 반올림 소수점 자리수

**Kind:** static property of [Constants](#)

**Constants.NODE\_TYPE**

캔버스 노드 타입 정의

**Kind:** static property of [Constants](#)

**Constants.SHAPES\_TYPE**

Shape 타입 정의

**Kind:** static property of [Constants](#)

**Constants.EDGE\_TYPE**

Edge 타입 정의

**Kind:** static property of [Constants](#)

**Constants.LABEL\_SUFFIX**

라벨 ID의 suffix 정의

**Kind:** static property of [Constants](#)

**Constants.LABEL\_EDITOR\_SUFFIX**

라벨 에디터 ID의 suffix 정의

**Kind:** static property of [Constants](#)

**Constants.FROM\_LABEL\_SUFFIX**

시작점 라벨 ID의 suffix 정의

**Kind:** static property of [Constants](#)

**Constants.TO\_LABEL\_SUFFIX**

끝점 라벨 ID의 suffix 정의

**Kind:** static property of [Constants](#)

**Constants.RUBBER\_BAND\_ID**

Rectangle 모양의 마우스 드래그 선택 박스 영역

**Kind:** static property of [Constants](#)

**Constants.RUBBER\_BAND\_TOLERANCE**

Rubber Band 허용 오차

**Kind:** static property of [Constants](#)

**Constants.GUIDE\_SUFFIX**

Move & Resize 용 가이드 ID 의 suffix 정의

**Kind:** static property of [Constants](#)

**Constants.COLLAPSE\_SUFFIX**

Collapse & Expand 용 가이드 ID의 suffix 정의

**Kind:** static property of [Constants](#)

**Constants.LOOPTYPE\_SUFFIX**

LoopType 용 가이드 ID의 suffix 정의

**Kind:** static property of [Constants](#)

**Constants.TASKTYPE\_SUFFIX**

TaskType 용 가이드 ID의 suffix 정의

**Kind:** static property of [Constants](#)

**Constants.INCLUSION\_SUFFIX**

TaskType 용 가이드 ID의 suffix 정의

**Kind:** static property of [Constants](#)

**Constants.STATUS\_SUFFIX**

STATUS 용 가이드 ID의 suffix 정의

**Kind:** static property of [Constants](#)

**Constants.EXCEPTIONTYPE\_SUFFIX**

EXCEPTIONTYPE 용 가이드 ID의 suffix 정의

**Kind:** static property of [Constants](#)

`Constants.MOVE_SNAP_SIZE`

Shape Move & Resize 시 이동 간격

**Kind:** static property of [Constants](#)

`Constants.DROP_OVER_BBOX_SUFFIX`

Edge 연결할때 Drop Over 가이드 ID의 suffix 정의

**Kind:** static property of [Constants](#)

`Constants.TERMINAL_SUFFIX`

Shape – Edge 와의 연결 포인트 터미널 ID의 suffix 정의

**Kind:** static property of [Constants](#)

`Constants.TERMINAL`

Shape – Edge 와의 연결 포인트

**Kind:** static property of [Constants](#)

`Constants.MARKER_TEMP_NODE`

마커 등록을 위한 임시 노드 아이디

**Kind:** static property of [Constants](#)

`Constants.PATTERN_TEMP_NODE`

패턴 등록을 위한 임시 노드 아이디

**Kind:** static property of [Constants](#)

`Constants.MARKER_DEFS_SUFFIX`

캔버스의 마커 데피니션 suffix 정의

**Kind:** static property of [Constants](#)

`Constants.ORIGINAL_NODE`

Shape에서 마커가 그려질 경우 원본 노드 suffix 정의

**Kind:** static property of [Constants](#)

`Constants.CONNECT_GUIDE_EVENT_AREA`

Element의 커넥트 가이드 이벤트 보정영역의 정의

**Kind:** static property of [Constants](#)

`Constants.CONNECT_GUIDE_SUFFIX`

Element 의 커넥트 가이드 suffix 정의

**Kind:** static property of [Constants](#)

## common.CurveUtil

**Kind:** static class of [common](#)

**Author:** [Seungpil Park \(mailto:sppark@uengine.org\)](mailto:Seungpil.Park@sppark@uengine.org)

- [.CurveUtil](#)
  - [new OG.common.CurveUtil\(\)](#)
  - [.CatmullRomSpline\(points\)](#) ⇒ Object
  - [.Bezier\(points\)](#) ⇒ Object
  - [.BSpline\(points, order\)](#) ⇒ Object

`new OG.common.CurveUtil()`

곡선(Curve) 알고리즘을 구현한 Javascript 클래스

`CurveUtil.CatmullRomSpline(points)` ⇒ Object

주어진 좌표 Array 에 대해 Cubic Catmull–Rom spline Curve 좌표를 계산하는 함수를 반환한다.  
모든 좌표를 지나는 커브를 계산한다.

**Kind:** static method of [CurveUtil](#)

**Returns:** Object – t 값에 의해 X, Y 좌표를 구하는 함수와 maxT 값을 반환

**Param Type** **Description**

points Array 좌표 Array (예, [[x1,y1], [x2,y2], [x3,y3], [x4,y4]])

## Example

```
var points = [[2, 2], [2, -2], [-2, 2], [-2, -2]],  
    cmRomSpline = OG.CurveUtil.CatmullRomSpline(points), t, curve = [];  
  
// t 는 0 ~ maxT 의 값으로, t 값의 증분값이 작을수록 세밀한 Curve 를 그린다.  
for(t = 0; t <= cmRomSpline.maxT; t += 0.1) {  
    curve.push([cmRomSpline.getX(t), cmRomSpline.getY(t)]);  
}
```

`CurveUtil.Bezier(points)` ⇒ Object

주어진 좌표 Array (좌표1, 콘트롤포인트1, 콘트롤포인트2, 좌표2 ...) 에 대해 Cubic Bezier Curve 좌표를 계산하는 함수를 반환한다.

Array 갯수는  $3 * K + 1$  이어야 한다.

예) 좌표1, 콘트롤포인트1, 콘트롤포인트2, 좌표2, 콘트롤포인트1, 콘트롤포인트2, 좌표3 ...

**Kind:** static method of [CurveUtil](#)

**Returns:** Object – t 값에 의해 X, Y 좌표를 구하는 함수와 maxT 값을 반환

**Param Type** **Description**

points Array 좌표 Array (예, [[x1,y1], [cp\_x1,cp\_y1], [cp\_x2,cp\_y2], [x2,y4]])

## Example

```
var points = [[2, 1], [1, 3], [-1, -1], [-2, 1]],  
    bezier = OG.CurveUtil.Bezier(points), t, curve = [];  
  
// t 는 0 ~ maxT 의 값으로, t 값의 증분값이 작을수록 세밀한 Curve 를 그린다.  
for(t = 0; t <= bezier.maxT; t += 0.1) {  
    curve.push([bezier.getX(t), bezier.getY(t)]);  
}
```

`CurveUtil.BSpline(points, order)` ⇒ Object

주어진 좌표 Array (시작좌표, 콘트롤포인트1, 콘트롤포인트2, ..., 끝좌표)에 대해 B-Spline Curve 좌표를 계산하는 함수를 반환한다.

**Kind:** static method of [CurveUtil](#)

**Returns:** Object – t 값에 의해 X, Y 좌표를 구하는 함수와 maxT 값을 반환

**Param Type** **Description**

points Array 좌표 Array (예, [[x1,y1], [x2,y2], [x3,y3], [x4,y4]])

order Number Order of the B-spline curve.

## Example

```
var points = [[2, 1], [1, 3], [-1, -1], [-2, 1]],  
    bspline = OG.CurveUtil.BSpline(points), t, curve = [];  
  
// t 는 0 ~ maxT 의 값으로, t 값의 증분값이 작을수록 세밀한 Curve 를 그린다.  
for(t = 0; t <= bspline.maxT; t += 0.1) {  
    curve.push([bspline.getX(t), bspline.getY(t)]);  
}
```

`common.HashMap`

**Kind:** static class of [common](#)

**Author:** [Seungpil Park \(mailto:sppark@uengine.org\)](mailto:Seungpil.Park@sppark@uengine.org)

- [.HashMap](#)

- [new OG.common.HashMap\(jsonObject\)](#)
- [.map](#) : Object
- [.put\(key, value\)](#)
- [.get\(key\)](#) ⇒ Object
- [.containsKey\(key\)](#) ⇒ Boolean
- [.containsValue\(value\)](#) ⇒ Boolean
- [.isEmpty\(\)](#) ⇒ Boolean
- [.clear\(\)](#)
- [.remove\(key\)](#)
- [.keys\(\)](#) ⇒ Array.<String>
- [.values\(\)](#) ⇒ Array.<Object>
- [.size\(\)](#) ⇒ Number
- [.toString\(\)](#) ⇒ String

```
new OG.common.HashMap(jsonObject)
```

HashMap 구현 Javascript 클래스

Param	Type	Description
jsonObject	Object	key:value 매핑 JSON 오브젝트

### Example

```
var map1 = new OG.common.HashMap({  
    'key1': 'value1',  
    'key2': 'value2'  
});  
  
console.log(map1.get('key1'));  
  
var map2 = new OG.common.HashMap();  
map2.put('key1', 'value1');  
map2.put('key2', 'value2');  
  
console.log(map2.get('key1'));
```

**hashMap.map** : Object

key:value 매핑 JSON 오브젝트

**Kind:** instance property of [HashMap](#)

**hashMap.put(key, value)**

key : value 를 매핑한다.

**Kind:** instance method of [HashMap](#)

**Param Type Description**

key String 키

value Object 값

**hashMap.get(key) ⇒ Object**

key 에 대한 value 를 반환한다.

**Kind:** instance method of [HashMap](#)

**Returns:** Object – 값

**Param Type Description**

key String 키

**hashMap.containsKey(key) ⇒ Boolean**

주어진 key 를 포함하는지 여부를 반환한다.

**Kind:** instance method of [HashMap](#)

**Param Type Description**

key String 키

`hashMap.containsKey(value) ⇒ Boolean`

주어진 value 를 포함하는지 여부를 반환한다.

**Kind:** instance method of [HashMap](#)

**Param Type Description**

value Object 값

`hashMap.isEmpty() ⇒ Boolean`

Empty 여부를 반환한다.

**Kind:** instance method of [HashMap](#)

`hashMap.clear()`

매핑정보를 클리어한다.

**Kind:** instance method of [HashMap](#)

`hashMap.remove(key)`

주어진 key 의 매핑정보를 삭제한다.

**Kind:** instance method of [HashMap](#)

**Param Type Description**

key String 키

`hashMap.keys() ⇒ Array.<String>`

key 목록을 반환한다.

**Kind:** instance method of [HashMap](#)

**Returns:** Array.<String> – 키목록

`hashMap.values() ⇒ Array.<Object>`

value 목록을 반환한다.

**Kind:** instance method of [HashMap](#)

**Returns:** Array.<Object> – 값목록

`hashMap.size() ⇒ Number`

매핑된 key:value 갯수를 반환한다.

**Kind:** instance method of [HashMap](#)

`hashMap.toString() ⇒ String`

객체 프라퍼티 정보를 JSON 스트링으로 반환한다.

**Kind:** instance method of [HashMap](#)

**Returns:** String – 프라퍼티 정보

## common.JSON

**Kind:** static class of [common](#)

**Author:** [Seungpil Park \(mailto:sppark@uengine.org\)](mailto:Seungpil.Park@sppark@uengine.org)

- [JSON](#)
  - [new OG.common.JSON\(\)](#)
  - [.encode](#) ⇒ String
  - [.decode](#) ⇒ Object
  - [.encodeDate\(d\)](#) ⇒ String

`new OG.common.JSON()`

Modified version of Douglas Crockford's json.js that doesn't mess with the Object prototype  
<http://www.json.org/js.html>

`jsoN.encode` ⇒ String

Encodes an Object, Array or other value

**Kind:** instance property of [JSON](#)

**Returns:** String – The JSON string

**Param** **Type**      **Description**

◦ Mixed The variable to encode

`jsoN.decode` ⇒ Object

Decodes (parses) a JSON string to an object. If the JSON is invalid, this function throws a SyntaxError unless the safe option is set.

**Kind:** instance property of [JSON](#)

**Returns:** Object – The resulting object

**Param** **Type**      **Description**

json    String The JSON string

`jsoN.encodeDate(d)` ⇒ String

Encodes a Date. This returns the actual string which is inserted into the JSON string as the literal expression. **The returned value includes enclosing double quotation marks.**

The default return format is "yyyy-mm-ddThh:mm:ss".

To override this:

```
OG.common.JSON.encodeDate = function(d) {
    return d.format('Y-m-d');
};
```

**Kind:** instance method of [JSON](#)

**Returns:** String – The string literal to use in a JSON string.

Param	Type	Description
-------	------	-------------

d	Date	The Date to encode
---	------	--------------------

## common.Util

**Kind:** static class of [common](#)

**Author:** [Seungpil Park \(mailto:sppark@uengine.org\)](mailto:Seungpil.Park@sppark@uengine.org)

- [Util](#)
  - [new OG.common.Util\(\)](#)
  - [.extend](#) ⇒ function
  - [.clone\(obj\)](#) ⇒ Object
  - [.round\(val\)](#) ⇒ Number
  - [.roundPrecision\(val, precision\)](#) ⇒ Number
  - [.roundGrid\(val, snapSize\)](#) ⇒ Number
  - [.apply\(obj, config, defaults\)](#) ⇒ Object

**new OG.common.Util()**

공통 유틸리티 Javascript 클래스

**Util.extend** ⇒ function

Extends one class to create a subclass and optionally overrides members with the passed literal. This method also adds the function "override()" to the subclass that can be used to override members of the class.

For example, to create a subclass of Ext GridPanel:

```
MyGridPanel = Ext.extend(Ext.grid.GridPanel, {
    constructor: function(config) {
```

```

//      Create configuration for this Grid.
var store = new Ext.data.Store({...});
var colModel = new Ext.grid.ColumnModel({...});

//      Create a new config object containing our computed properties
//      *plus* whatever was in the config parameter.
config = Ext.apply({
store: store,
colModel: colModel
}, config);

MyGridPanel.superclass.constructor.call(this, config);

//      Your postprocessing here
},

yourMethod: function() {
// etc.
}
});
</code></pre>

```

This function also supports a 3-argument call in which the subclass's constructor is passed as an argument. In this form, the parameters are as follows:

- **subclass : Function**  
The subclass constructor.
- **superclass : Function**  
The constructor of class being extended
- **overrides : Object**  
A literal with members which are copied into the subclass's prototype, and are therefore shared among all instances of the new class.

Kind: static property of [Util](#)

Returns: function - The subclass constructor from the overrides parameter, or a generated one if not provided.

Param	Type	Description
superclass	function	The constructor of class being extended.

A literal with members which are copied into the subclass's prototype, and are therefore shared between all instances of the new class.

overrides Object This may contain a special member named constructor. This is used to define the constructor of the new class, and is returned. If this property is *not* specified, a constructor is generated and returned which just calls the superclass's constructor passing on its parameters.

It is essential that you call the superclass constructor in any provided constructor. See example code.

Util.clone(obj) ⇒ Object

Object 를 복사한다.

Kind: static method of [Util](#)

Returns: Object - 복사된 Object

Param Type Description  
obj Object 복사할 Object

Util.round(val) ⇒ Number

디폴트로 지정된 소수점 자리수로 Round 한 값을 반환한다.

Kind: static method of [Util](#)

Returns: Number - 지정한 소수점 자리수에 따른 반올림 값

Param Type Description  
val Number 반올림할 값

```
Util.roundPrecision(val, precision) => Number
```

입력된 숫자값을 지정된 소수점 자릿수로 Round해서 값을 리턴한다.

Kind: static method of [Util](#)

Returns: Number - 지정한 소수점 자리수에 따른 반올림 값

Param	Type	Description
val	Number	반올림할 값
precision	Number	소수점 자리수

Example

```
OG.Util.roundPrecision(300.12345678, 3);  
Result ) 300.123
```

```
Util.roundGrid(val, snapSize) => Number
```

Shape Move & Resize 이동 간격으로 Round 한 값을 반환한다.

Kind: static method of [Util](#)

Returns: Number - 지정한 간격으로 반올림 값

Param	Type	Description
val	Number	반올림할 값
snapSize	Number	이동간격

```
Util.apply(obj, config, defaults) => Object
```

Copies all the properties of config to obj.

Kind: static method of [Util](#)

Returns: Object - returns obj

Param	Type	Description
obj	Object	The receiver of the properties
config	Object	The source of the properties
defaults	Object	A different object that will also be applied for default values

OG.geometry : object

Kind: static namespace of [OG](#)

- [.geometry](#) : object
  - [.BezierCurve](#) ⇐ [PolyLine](#)
    - [new OG.geometry.BezierCurve\(controlPoints\)](#)
    - [.controlPoints](#) : [Array.<Coordinate>](#)
    - [.vertices](#) : [Array.<Coordinate>](#)
    - [.TYPE](#) : Number
    - [.IS\\_CLOSED](#) : Boolean
    - [.style](#) : [Style](#)
    - [.boundary](#) : [Envelope](#)
    - [.getControlPoints\(\)](#) ⇒ [Array.<Coordinate>](#)
    - [.getVertices\(\)](#) ⇒ [Array.<Coordinate>](#)
    - [.move\(offsetX, offsetY\)](#) ⇒ [Geometry](#)
    - [.resize\(upper, lower, left, right\)](#) ⇒ [Geometry](#)

- [.rotate\(angle, origin\)](#) ⇒ [Geometry](#)
- [.toString\(\)](#) ⇒ String
- [.angleBetweenPoints\(prev, next\)](#) ⇒ Number
- [.isRightAngleBetweenPoints\(prev, next\)](#) ⇒ Object
- [.angleBetweenThreePoints\(prev, next\)](#) ⇒ Number
- [.isEqual\(\\_geometry\)](#) ⇒ Boolean
- [.isContains\(\\_geometry\)](#) ⇒ Boolean
- [.isWithin\(\\_geometry\)](#) ⇒ Boolean
- [.getBoundary\(\)](#) ⇒ [Envelope](#)
- [.getCentroid\(\)](#) ⇒ [Coordinate](#)
- [.minDistance\(\\_coordinate\)](#) ⇒ Number
- [.distance\(\\_geometry\)](#) ⇒ Number
- [.getLength\(\)](#) ⇒ Number
- [.moveCentroid\(중심\)](#)
- [.resizeBox\(width, height\)](#) ⇒ [Geometry](#)
- [.fitToBoundary\(envelope\)](#) ⇒ [Geometry](#)
- [.convertCoordinate\(coordinate\)](#) ⇒ [Coordinate](#)
- [.distanceToLine\(p, line\)](#) ⇒ Number
- [.distanceLineToLine\(line1, line2\)](#) ⇒ Number
- [.intersectToLine\(line\)](#) ⇒ [Array.<Coordinate>](#)
- [.shortestIntersectToLine\(line\)](#) ⇒ [Array.<Coordinate>](#)
- [.intersectLineToLine\(line1, line2, extension\)](#) ⇒ [Coordinate](#)
- [.intersectCircleToLine\(center, radius, from, to\)](#) ⇒ [Array.<Coordinate>](#)
- [.intersectPointToLine\(p, line\)](#) ⇒ [Coordinate](#)
- [.getPercentageDistanceFromPoint\(\\_coordinate\)](#) ⇒ Object
- [.isContainsPoint\(\\_coordinate\)](#) ⇒ boolean

- [`.getPointFromPercentageDistance\(pXpY\)`](#) ⇒ [`Coordinate`](#)
- [`.getParallelLine\(from, to, distance\)`](#) ⇒ [`Array.<Coordinate>`](#)
- [`.getParallelPath\(line, distance\)`](#)
- [`.reset\(\)`](#)
- [`.Circle`](#) ≈ [`Ellipse`](#)
  - [`new OG.geometry.Circle\(center, radius\)`](#)
  - [`.vertices`](#) : [`Array.<Coordinate>`](#)
  - [`.TYPE`](#) : Number
  - [`.IS\_CLOSED`](#) : Boolean
  - [`.style`](#) : [`Style`](#)
  - [`.boundary`](#) : [`Envelope`](#)
  - [`.getVertices\(\)`](#) ⇒ [`Array.<Coordinate>`](#)
  - [`.getControlPoints\(\)`](#) ⇒ [`Array.<Coordinate>`](#)
  - [`.getLength\(\)`](#) ⇒ Number
  - [`.toString\(\)`](#) ⇒ String
  - [`.move\(offsetX, offsetY\)`](#) ⇒ [`Geometry`](#)
  - [`.resize\(upper, lower, left, right\)`](#) ⇒ [`Geometry`](#)
  - [`.rotate\(angle, origin\)`](#) ⇒ [`Geometry`](#)
  - [`.angleBetweenPoints\(prev, next\)`](#) ⇒ Number
  - [`.isRightAngleBetweenPoints\(prev, next\)`](#) ⇒ Object
  - [`.angleBetweenThreePoints\(prev, next\)`](#) ⇒ Number
  - [`.isEqual\(\_geometry\)`](#) ⇒ Boolean
  - [`.isContains\(\_geometry\)`](#) ⇒ Boolean
  - [`.isWithin\(\_geometry\)`](#) ⇒ Boolean

- [.getBoundary\(\)](#) ⇒ [Envelope](#)
- [.getCentroid\(\)](#) ⇒ [Coordinate](#)
- [.minDistance\(\\_coordinate\)](#) ⇒ Number
- [.distance\(\\_geometry\)](#) ⇒ Number
- [.moveCentroid\(중심\)](#)
- [.resizeBox\(width, height\)](#) ⇒ [Geometry](#)
- [.fitToBoundary\(envelope\)](#) ⇒ [Geometry](#)
- [.convertCoordinate\(coordinate\)](#) ⇒ [Coordinate](#)
- [.distanceToLine\(p, line\)](#) ⇒ Number
- [.distanceLineToLine\(line1, line2\)](#) ⇒ Number
- [.intersectToLine\(line\)](#) ⇒ [Array.<Coordinate>](#)
- [.shortestIntersectToLine\(line\)](#) ⇒ [Array.<Coordinate>](#)
- [.intersectLineToLine\(line1, line2, extension\)](#) ⇒ [Coordinate](#)
- [.intersectCircleToLine\(center, radius, from, to\)](#) ⇒ [Array.<Coordinate>](#)
- [.intersectPointToLine\(p, line\)](#) ⇒ [Coordinate](#)
- [.getPercentageDistanceFromPoint\(\\_coordinate\)](#) ⇒ Object
- [.isContainsPoint\(\\_coordinate\)](#) ⇒ boolean
- [.getPointFromPercentageDistance\(pXpY\)](#) ⇒ [Coordinate](#)
- [.getParallelLine\(from, to, distance\)](#) ⇒ [Array.<Coordinate>](#)
- [.getParallelPath\(line, distance\)](#)
- [.reset\(\)](#)

- [.Coordinate](#)

- [new OG.geometry.Coordinate\(x, y\)](#)
- [.x](#) : Number

- [.y](#) : Number
  - [.distance\(coordinate\)](#) ⇒ Number
  - [.move\(offsetX, offsetY\)](#) ⇒ [Coordinate](#)
  - [.rotate\(angle, origin\)](#) ⇒ [Coordinate](#)
  - [.equals\(coordinate\)](#) ⇒ Boolean
  - [.toString\(\)](#) ⇒ String
- 
- [.Curve](#) ≈ [PolyLine](#)
    - [new OG.geometry.Curve\(controlPoints\)](#)
    - [.vertices](#) : [Array.<Coordinate>](#)
    - [.TYPE](#) : Number
    - [.IS\\_CLOSED](#) : Boolean
    - [.style](#) : [Style](#)
    - [.boundary](#) : [Envelope](#)
    - [.getControlPoints\(\)](#) ⇒ [Array.<Coordinate>](#)
    - [.getVertices\(\)](#) ⇒ [Array.<Coordinate>](#)
    - [.move\(offsetX, offsetY\)](#) ⇒ [Geometry](#)
    - [.resize\(upper, lower, left, right\)](#) ⇒ [Geometry](#)
    - [.rotate\(angle, origin\)](#) ⇒ [Geometry](#)
    - [.toString\(\)](#) ⇒ String
    - [.angleBetweenPoints\(prev, next\)](#) ⇒ Number
    - [.isRightAngleBetweenPoints\(prev, next\)](#) ⇒ Object
    - [.angleBetweenThreePoints\(prev, next\)](#) ⇒ Number
    - [.equals\(geometry\)](#) ⇒ Boolean
    - [.contains\(geometry\)](#) ⇒ Boolean
    - [.within\(geometry\)](#) ⇒ Boolean

- [.getBoundary\(\)](#) ⇒ [Envelope](#)
  - [.getCentroid\(\)](#) ⇒ [Coordinate](#)
  - [.minDistance\(\\_coordinate\)](#) ⇒ Number
  - [.distance\(\\_geometry\)](#) ⇒ Number
  - [.getLength\(\)](#) ⇒ Number
  - [.moveCentroid\(중심\)](#)
  - [.resizeBox\(width, height\)](#) ⇒ [Geometry](#)
  - [.fitToBoundary\(envelope\)](#) ⇒ [Geometry](#)
  - [.convertCoordinate\(coordinate\)](#) ⇒ [Coordinate](#)
  - [.distanceToLine\(p, line\)](#) ⇒ Number
  - [.distanceLineToLine\(line1, line2\)](#) ⇒ Number
  - [.intersectToLine\(line\)](#) ⇒ [Array.<Coordinate>](#)
  - [.shortestIntersectToLine\(line\)](#) ⇒ [Array.<Coordinate>](#)
  - [.intersectLineToLine\(line1, line2, extension\)](#) ⇒ [Coordinate](#)
  - [.intersectCircleToLine\(center, radius, from, to\)](#) ⇒ [Array.<Coordinate>](#)
  - [.intersectPointToLine\(p, line\)](#) ⇒ [Coordinate](#)
  - [.getPercentageDistanceFromPoint\(\\_coordinate\)](#) ⇒ Object
  - [.isContainsPoint\(\\_coordinate\)](#) ⇒ boolean
  - [.getPointFromPercentageDistance\(pXpY\)](#) ⇒ [Coordinate](#)
  - [.getParallelLine\(from, to, distance\)](#) ⇒ [Array.<Coordinate>](#)
  - [.getParallelPath\(line, distance\)](#)
  - [.reset\(\)](#)
- [.Ellipse](#) ⇐ [Curve](#)
- [new OG.geometry.Ellipse\(center, radiusX, radiusY, angle\)](#)

- [.vertices](#) : [Array.<Coordinate>](#)
- [.TYPE](#) : Number
- [.IS\\_CLOSED](#) : Boolean
- [.style](#) : [Style](#)
- [.boundary](#) : [Envelope](#)
- [.getControlPoints\(\)](#) ⇒ [Array.<Coordinate>](#)
- [.getVertices\(\)](#) ⇒ [Array.<Coordinate>](#)
- [.toString\(\)](#) ⇒ String
- [.move\(offsetX, offsetY\)](#) ⇒ [Geometry](#)
- [.resize\(upper, lower, left, right\)](#) ⇒ [Geometry](#)
- [.rotate\(angle, origin\)](#) ⇒ [Geometry](#)
- [.angleBetweenPoints\(prev, next\)](#) ⇒ Number
- [.isRightAngleBetweenPoints\(prev, next\)](#) ⇒ Object
- [.angleBetweenThreePoints\(prev, next\)](#) ⇒ Number
- [.isEqual\(\\_geometry\)](#) ⇒ Boolean
- [.isContains\(\\_geometry\)](#) ⇒ Boolean
- [.isWithin\(\\_geometry\)](#) ⇒ Boolean
- [.getBoundary\(\)](#) ⇒ [Envelope](#)
- [.getCentroid\(\)](#) ⇒ [Coordinate](#)
- [.minDistance\(\\_coordinate\)](#) ⇒ Number
- [.distance\(\\_geometry\)](#) ⇒ Number
- [.getLength\(\)](#) ⇒ Number
- [.moveCentroid\(중심\)](#)
- [.resizeBox\(width, height\)](#) ⇒ [Geometry](#)
- [.fitToBoundary\(envelope\)](#) ⇒ [Geometry](#)
- [.convertCoordinate\(coordinate\)](#) ⇒ [Coordinate](#)

- [`.distanceToLine\(p, line\)`](#) ⇒ Number
- [`.distanceLineToLine\(line1, line2\)`](#) ⇒ Number
- [`.intersectToLine\(line\)`](#) ⇒ [`Array.<Coordinate>`](#)
- [`.shortestIntersectToLine\(line\)`](#) ⇒ [`Array.<Coordinate>`](#)
- [`.intersectLineToLine\(line1, line2, extension\)`](#) ⇒ [`Coordinate`](#)
- [`.intersectCircleToLine\(center, radius, from, to\)`](#) ⇒ [`Array.<Coordinate>`](#)
- [`.intersectPointToLine\(p, line\)`](#) ⇒ [`Coordinate`](#)
- [`.getPercentageDistanceFromPoint\(\_coordinate\)`](#) ⇒ Object
- [`.isContainsPoint\(\_coordinate\)`](#) ⇒ boolean
- [`.getPointFromPercentageDistance\(pXpY\)`](#) ⇒ [`Coordinate`](#)
- [`.getParallelLine\(from, to, distance\)`](#) ⇒ [`Array.<Coordinate>`](#)
- [`.getParallelPath\(line, distance\)`](#)
- [`.reset\(\)`](#)

- [.Envelope](#)

- [`new OG.geometry.Envelope\(upperLeft, width, height\)`](#)
- [`.getUpperLeft\(\)`](#) ⇒ [`Coordinate`](#)
- [`.setUpperLeft\(upperLeft\)`](#)
- [`.getUpperRight\(\)`](#) ⇒ [`Coordinate`](#)
- [`.getLowerRight\(\)`](#) ⇒ [`Coordinate`](#)
- [`.getLowerLeft\(\)`](#) ⇒ [`Coordinate`](#)
- [`.getLeftCenter\(\)`](#) ⇒ [`Coordinate`](#)
- [`.getUpperCenter\(\)`](#) ⇒ [`Coordinate`](#)
- [`.getRightCenter\(\)`](#) ⇒ [`Coordinate`](#)
- [`.getLowerCenter\(\)`](#) ⇒ [`Coordinate`](#)

- [.getCentroid\(\)](#) ⇒ [Coordinate](#)
- [.setCentroid\(centroid\)](#)
- [.getWidth\(\)](#) ⇒ Number
- [.setWidth\(width\)](#)
- [.getHeight\(\)](#) ⇒ Number
- [.setHeight\(height\)](#)
- [.getVertices\(\)](#) ⇒ [Array.<Coordinate>](#)
- [.isContains\(coordinate\)](#) ⇒ Boolean
- [.isContainsAll\(coordinateArray\)](#) ⇒ Boolean
- [.getHowManyContains\(coordinateArray\)](#) ⇒ Boolean
- [.isContainsOnce\(coordinateArray\)](#) ⇒ Boolean
- [.move\(offsetX, offsetY\)](#) ⇒ [Envelope](#)
- [.resize\(upper, lower, left, right\)](#) ⇒ [Envelope](#)
- [.equals\(Envelope\)](#) ⇒ Boolean
- [.toString\(\)](#) ⇒ String

- [.Geometry](#)

- [new OG.geometry.Geometry\(\)](#)
- [.TYPE](#) : Number
- [.IS\\_CLOSED](#) : Boolean
- [.style](#) : [Style](#)
- [.boundary](#) : [Envelope](#)
- [.equals\(\\_geometry\)](#) ⇒ Boolean
- [.isContains\(\\_geometry\)](#) ⇒ Boolean
- [.isWithin\(\\_geometry\)](#) ⇒ Boolean

- [`.getBoundary\(\)`](#) ⇒ [Envelope](#)
- [`.getCentroid\(\)`](#) ⇒ [Coordinate](#)
- [`.getVertices\(\)`](#) ⇒ [Array.<Coordinate>](#)
- [`.minDistance\(\_coordinate\)`](#) ⇒ Number
- [`.distance\(\_geometry\)`](#) ⇒ Number
- [`.getLength\(\)`](#) ⇒ Number
- [`.move\(offsetX, offsetY\)`](#) ⇒ [Geometry](#)
- [`.moveCentroid\(중심\)`](#)
- [`.resize\(upper, lower, left, right\)`](#) ⇒ [Geometry](#)
- [`.resizeBox\(width, height\)`](#) ⇒ [Geometry](#)
- [`.rotate\(angle, origin\)`](#) ⇒ [Geometry](#)
- [`.fitToBoundary\(envelope\)`](#) ⇒ [Geometry](#)
- [`.convertCoordinate\(coordinate\)`](#) ⇒ [Coordinate](#)
- [`.distanceToLine\(p, line\)`](#) ⇒ Number
- [`.distanceLineToLine\(line1, line2\)`](#) ⇒ Number
- [`.intersectToLine\(line\)`](#) ⇒ [Array.<Coordinate>](#)
- [`.shortestIntersectToLine\(line\)`](#) ⇒ [Array.<Coordinate>](#)
- [`.intersectLineToLine\(line1, line2, extension\)`](#) ⇒ [Coordinate](#)
- [`.intersectCircleToLine\(center, radius, from, to\)`](#) ⇒ [Array.<Coordinate>](#)
- [`.intersectPointToLine\(p, line\)`](#) ⇒ [Coordinate](#)
- [`.getPercentageDistanceFromPoint\(\_coordinate\)`](#) ⇒ Object
- [`.isContainsPoint\(\_coordinate\)`](#) ⇒ boolean
- [`.getPointFromPercentageDistance\(pXpY\)`](#) ⇒ [Coordinate](#)
- [`.getParallelLine\(from, to, distance\)`](#) ⇒ [Array.<Coordinate>](#)
- [`.getParallelPath\(line, distance\)`](#)
- [`.reset\(\)`](#)

- [.GeometryCollection](#) ⇐ [Geometry](#)
  - [new OG.geometry.GeometryCollection\(geometries\)](#)
  - [.geometries](#) : [Array.<Geometry>](#)
  - [.TYPE](#) : Number
  - [.IS\\_CLOSED](#) : Boolean
  - [.style](#) : [Style](#)
  - [.boundary](#) : [Envelope](#)
  - [.toString\(\)](#) ⇒ String
  - [.isEquals\(\\_geometry\)](#) ⇒ Boolean
  - [.isContains\(\\_geometry\)](#) ⇒ Boolean
  - [.isWithin\(\\_geometry\)](#) ⇒ Boolean
  - [.getBoundary\(\)](#) ⇒ [Envelope](#)
  - [.getCentroid\(\)](#) ⇒ [Coordinate](#)
  - [.getVertices\(\)](#) ⇒ [Array.<Coordinate>](#)
  - [.minDistance\(\\_coordinate\)](#) ⇒ Number
  - [.distance\(\\_geometry\)](#) ⇒ Number
  - [.getLength\(\)](#) ⇒ Number
  - [.move\(offsetX, offsetY\)](#) ⇒ [Geometry](#)
  - [.moveCentroid\(중심\)](#)
  - [.resize\(upper, lower, left, right\)](#) ⇒ [Geometry](#)
  - [.resizeBox\(width, height\)](#) ⇒ [Geometry](#)
  - [.rotate\(angle, origin\)](#) ⇒ [Geometry](#)
  - [.fitToBoundary\(envelope\)](#) ⇒ [Geometry](#)
  - [.convertCoordinate\(coordinate\)](#) ⇒ [Coordinate](#)

- [`.distanceToLine\(p, line\)`](#) ⇒ Number
- [`.distanceLineToLine\(line1, line2\)`](#) ⇒ Number
- [`.intersectToLine\(line\)`](#) ⇒ `Array.<Coordinate>`
- [`.shortestIntersectToLine\(line\)`](#) ⇒ `Array.<Coordinate>`
- [`.intersectLineToLine\(line1, line2, extension\)`](#) ⇒ `Coordinate`
- [`.intersectCircleToLine\(center, radius, from, to\)`](#) ⇒ `Array.<Coordinate>`
- [`.intersectPointToLine\(p, line\)`](#) ⇒ `Coordinate`
- [`.getPercentageDistanceFromPoint\(\_coordinate\)`](#) ⇒ Object
- [`.isContainsPoint\(\_coordinate\)`](#) ⇒ boolean
- [`.getPointFromPercentageDistance\(pXpY\)`](#) ⇒ `Coordinate`
- [`.getParallelLine\(from, to, distance\)`](#) ⇒ `Array.<Coordinate>`
- [`.getParallelPath\(line, distance\)`](#)
- [`.reset\(\)`](#)

◦ [`.Line`](#) ⇐ [`PolyLine`](#)

- [`new OG.geometry.Line\(from, to\)`](#)
- [`.vertices`](#) : `Array.<Coordinate>`
- [`.TYPE`](#) : Number
- [`.IS\_CLOSED`](#) : Boolean
- [`.style`](#) : [`Style`](#)
- [`.boundary`](#) : [`Envelope`](#)
- [`.getVertices\(\)`](#) ⇒ `Array.<Coordinate>`
- [`.move\(offsetX, offsetY\)`](#) ⇒ [`Geometry`](#)
- [`.resize\(upper, lower, left, right\)`](#) ⇒ [`Geometry`](#)
- [`.rotate\(angle, origin\)`](#) ⇒ [`Geometry`](#)

- [.toString\(\)](#) ⇒ String
- [.angleBetweenPoints\(prev, next\)](#) ⇒ Number
- [.isRightAngleBetweenPoints\(prev, next\)](#) ⇒ Object
- [.angleBetweenThreePoints\(prev, next\)](#) ⇒ Number
- [.isEqual\(\\_geometry\)](#) ⇒ Boolean
- [.isContains\(\\_geometry\)](#) ⇒ Boolean
- [.isWithin\(\\_geometry\)](#) ⇒ Boolean
- [.getBoundary\(\)](#) ⇒ Envelope
- [.getCentroid\(\)](#) ⇒ Coordinate
- [.minDistance\(\\_coordinate\)](#) ⇒ Number
- [.distance\(\\_geometry\)](#) ⇒ Number
- [.getLength\(\)](#) ⇒ Number
- [.moveCentroid\(중심\)](#)
- [.resizeBox\(width, height\)](#) ⇒ Geometry
- [.fitToBoundary\(envelope\)](#) ⇒ Geometry
- [.convertCoordinate\(coordinate\)](#) ⇒ Coordinate
- [.distanceToLine\(p, line\)](#) ⇒ Number
- [.distanceLineToLine\(line1, line2\)](#) ⇒ Number
- [.intersectToLine\(line\)](#) ⇒ Array.<Coordinate>
- [.shortestIntersectToLine\(line\)](#) ⇒ Array.<Coordinate>
- [.intersectLineToLine\(line1, line2, extension\)](#) ⇒ Coordinate
- [.intersectCircleToLine\(center, radius, from, to\)](#) ⇒ Array.<Coordinate>
- [.intersectPointToLine\(p, line\)](#) ⇒ Coordinate
- [.getPercentageDistanceFromPoint\(\\_coordinate\)](#) ⇒ Object
- [.isContainsPoint\(\\_coordinate\)](#) ⇒ boolean
- [.getPointFromPercentageDistance\(pXpY\)](#) ⇒ Coordinate

- [.getParallelLine\(from, to, distance\)](#) ⇒ [Array.<Coordinate>](#)
- [.getParallelPath\(line, distance\)](#)
- [.reset\(\)](#)
- [.Point](#) ⇐ [Geometry](#)
  - [new OG.geometry.Point\(coordinate\)](#)
  - [.coordinate](#) : [Coordinate](#)
  - [.vertices](#) : [Array.<Coordinate>](#)
  - [.TYPE](#) : Number
  - [.IS\\_CLOSED](#) : Boolean
  - [.style](#) : [Style](#)
  - [.boundary](#) : [Envelope](#)
  - [.toString\(\)](#) ⇒ String
  - [.isEquals\(\\_geometry\)](#) ⇒ Boolean
  - [.isContains\(\\_geometry\)](#) ⇒ Boolean
  - [.isWithin\(\\_geometry\)](#) ⇒ Boolean
  - [.getBoundary\(\)](#) ⇒ [Envelope](#)
  - [.getCentroid\(\)](#) ⇒ [Coordinate](#)
  - [.getVertices\(\)](#) ⇒ [Array.<Coordinate>](#)
  - [.minDistance\(\\_coordinate\)](#) ⇒ Number
  - [.distance\(\\_geometry\)](#) ⇒ Number
  - [.getLength\(\)](#) ⇒ Number
  - [.move\(offsetX, offsetY\)](#) ⇒ [Geometry](#)
  - [.moveCentroid\(중심\)](#)
  - [.resize\(upper, lower, left, right\)](#) ⇒ [Geometry](#)

- [.resizeBox\(width, height\)](#) ⇒ [Geometry](#)
- [.rotate\(angle, origin\)](#) ⇒ [Geometry](#)
- [.fitToBoundary\(envelope\)](#) ⇒ [Geometry](#)
- [.convertCoordinate\(coordinate\)](#) ⇒ [Coordinate](#)
- [.distanceToLine\(p, line\)](#) ⇒ Number
- [.distanceLineToLine\(line1, line2\)](#) ⇒ Number
- [.intersectToLine\(line\)](#) ⇒ [Array.<Coordinate>](#)
- [.shortestIntersectToLine\(line\)](#) ⇒ [Array.<Coordinate>](#)
- [.intersectLineToLine\(line1, line2, extension\)](#) ⇒ [Coordinate](#)
- [.intersectCircleToLine\(center, radius, from, to\)](#) ⇒ [Array.<Coordinate>](#)
- [.intersectPointToLine\(p, line\)](#) ⇒ [Coordinate](#)
- [.getPercentageDistanceFromPoint\(\\_coordinate\)](#) ⇒ Object
- [.isContainsPoint\(\\_coordinate\)](#) ⇒ boolean
- [.getPointFromPercentageDistance\(pXpY\)](#) ⇒ [Coordinate](#)
- [.getParallelLine\(from, to, distance\)](#) ⇒ [Array.<Coordinate>](#)
- [.getParallelPath\(line, distance\)](#)
- [.reset\(\)](#)

- [.PolyLine](#) ⇌ [Geometry](#)

- [new OG.geometry.PolyLine\(vertices\)](#)
- [.vertices](#) : [Array.<Coordinate>](#)
- [.TYPE](#) : Number
- [.IS\\_CLOSED](#) : Boolean
- [.style](#) : [Style](#)
- [.boundary](#) : [Envelope](#)

- [.toString\(\)](#) ⇒ String
- [.angleBetweenPoints\(prev, next\)](#) ⇒ Number
- [.isRightAngleBetweenPoints\(prev, next\)](#) ⇒ Object
- [.angleBetweenThreePoints\(prev, next\)](#) ⇒ Number
- [.isEqual\(\\_geometry\)](#) ⇒ Boolean
- [.isContains\(\\_geometry\)](#) ⇒ Boolean
- [.isWithin\(\\_geometry\)](#) ⇒ Boolean
- [.getBoundary\(\)](#) ⇒ Envelope
- [.getCentroid\(\)](#) ⇒ Coordinate
- [.getVertices\(\)](#) ⇒ Array.<Coordinate>
- [.minDistance\(\\_coordinate\)](#) ⇒ Number
- [.distance\(\\_geometry\)](#) ⇒ Number
- [.getLength\(\)](#) ⇒ Number
- [.move\(offsetX, offsetY\)](#) ⇒ Geometry
- [.moveCentroid\(중심\)](#)
- [.resize\(upper, lower, left, right\)](#) ⇒ Geometry
- [.resizeBox\(width, height\)](#) ⇒ Geometry
- [.rotate\(angle, origin\)](#) ⇒ Geometry
- [.fitToBoundary\(envelope\)](#) ⇒ Geometry
- [.convertCoordinate\(coordinate\)](#) ⇒ Coordinate
- [.distanceToLine\(p, line\)](#) ⇒ Number
- [.distanceLineToLine\(line1, line2\)](#) ⇒ Number
- [.intersectToLine\(line\)](#) ⇒ Array.<Coordinate>
- [.shortestIntersectToLine\(line\)](#) ⇒ Array.<Coordinate>
- [.intersectLineToLine\(line1, line2, extension\)](#) ⇒ Coordinate
- [.intersectCircleToLine\(center, radius, from, to\)](#) ⇒ Array.<Coordinate>

- [.intersectPointToLine\(p, line\)](#) ⇒ [Coordinate](#)
  - [.getPercentageDistanceFromPoint\(\\_coordinate\)](#) ⇒ Object
  - [.isContainsPoint\(\\_coordinate\)](#) ⇒ boolean
  - [.getPointFromPercentageDistance\(pXpY\)](#) ⇒ [Coordinate](#)
  - [.getParallelLine\(from, to, distance\)](#) ⇒ [Array.<Coordinate>](#)
  - [.getParallelPath\(line, distance\)](#)
  - [.reset\(\)](#)
- [.Polygon](#) ⇐ [PolyLine](#)
  - [new OG.geometry.Polygon\(vertices\)](#)
  - [.vertices](#) : [Array.<Coordinate>](#)
  - [.TYPE](#) : Number
  - [.IS\\_CLOSED](#) : Boolean
  - [.style](#) : [Style](#)
  - [.boundary](#) : [Envelope](#)
  - [.getVertices\(\)](#) ⇒ [Array.<Coordinate>](#)
  - [.move\(offsetX, offsetY\)](#) ⇒ [Geometry](#)
  - [.resize\(upper, lower, left, right\)](#) ⇒ [Geometry](#)
  - [.rotate\(angle, origin\)](#) ⇒ [Geometry](#)
  - [.toString\(\)](#) ⇒ String
  - [.angleBetweenPoints\(prev, next\)](#) ⇒ Number
  - [.isRightAngleBetweenPoints\(prev, next\)](#) ⇒ Object
  - [.angleBetweenThreePoints\(prev, next\)](#) ⇒ Number
  - [.equals\(\\_geometry\)](#) ⇒ Boolean
  - [.isContains\(\\_geometry\)](#) ⇒ Boolean

- [.isWithin\(\\_geometry\)](#) ⇒ Boolean
- [.getBoundary\(\)](#) ⇒ [Envelope](#)
- [.getCentroid\(\)](#) ⇒ [Coordinate](#)
- [.minDistance\(\\_coordinate\)](#) ⇒ Number
- [.distance\(\\_geometry\)](#) ⇒ Number
- [.getLength\(\)](#) ⇒ Number
- [.moveCentroid\(중심\)](#)
- [.resizeBox\(width, height\)](#) ⇒ [Geometry](#)
- [.fitToBoundary\(envelope\)](#) ⇒ [Geometry](#)
- [.convertCoordinate\(coordinate\)](#) ⇒ [Coordinate](#)
- [.distanceToLine\(p, line\)](#) ⇒ Number
- [.distanceLineToLine\(line1, line2\)](#) ⇒ Number
- [.intersectToLine\(line\)](#) ⇒ [Array.<Coordinate>](#)
- [.shortestIntersectToLine\(line\)](#) ⇒ [Array.<Coordinate>](#)
- [.intersectLineToLine\(line1, line2, extension\)](#) ⇒ [Coordinate](#)
- [.intersectCircleToLine\(center, radius, from, to\)](#) ⇒ [Array.<Coordinate>](#)
- [.intersectPointToLine\(p, line\)](#) ⇒ [Coordinate](#)
- [.getPercentageDistanceFromPoint\(\\_coordinate\)](#) ⇒ Object
- [.isContainsPoint\(\\_coordinate\)](#) ⇒ boolean
- [.getPointFromPercentageDistance\(pXpY\)](#) ⇒ [Coordinate](#)
- [.getParallelLine\(from, to, distance\)](#) ⇒ [Array.<Coordinate>](#)
- [.getParallelPath\(line, distance\)](#)
- [.reset\(\)](#)

- [.Rectangle](#) ⇌ [Polygon](#)

- [new OG.geometry.Rectangle\(upperLeft, width, height\)](#)
- [.vertices : Array.<Coordinate>](#)
- [.TYPE : Number](#)
- [.IS\\_CLOSED : Boolean](#)
- [.style : Style](#)
- [.boundary : Envelope](#)
- [.getVertices\(\) ⇒ Array.<Coordinate>](#)
- [.move\(offsetX, offsetY\) ⇒ Geometry](#)
- [.resize\(upper, lower, left, right\) ⇒ Geometry](#)
- [.rotate\(angle, origin\) ⇒ Geometry](#)
- [.toString\(\) ⇒ String](#)
- [.angleBetweenPoints\(prev, next\) ⇒ Number](#)
- [.isRightAngleBetweenPoints\(prev, next\) ⇒ Object](#)
- [.angleBetweenThreePoints\(prev, next\) ⇒ Number](#)
- [.isEqual\(\\_geometry\) ⇒ Boolean](#)
- [.isContains\(\\_geometry\) ⇒ Boolean](#)
- [.isWithin\(\\_geometry\) ⇒ Boolean](#)
- [.getBoundary\(\) ⇒ Envelope](#)
- [.getCentroid\(\) ⇒ Coordinate](#)
- [.minDistance\(\\_coordinate\) ⇒ Number](#)
- [.distance\(\\_geometry\) ⇒ Number](#)
- [.getLength\(\) ⇒ Number](#)
- [.moveCentroid\(중심\)](#)
- [.resizeBox\(width, height\) ⇒ Geometry](#)
- [.fitToBoundary\(envelope\) ⇒ Geometry](#)
- [.convertCoordinate\(coordinate\) ⇒ Coordinate](#)

- [.distanceToLine\(p, line\)](#) ⇒ Number
  - [.distanceLineToLine\(line1, line2\)](#) ⇒ Number
  - [.intersectToLine\(line\)](#) ⇒ [Array.<Coordinate>](#)
  - [.shortestIntersectToLine\(line\)](#) ⇒ [Array.<Coordinate>](#)
  - [.intersectLineToLine\(line1, line2, extension\)](#) ⇒ [Coordinate](#)
  - [.intersectCircleToLine\(center, radius, from, to\)](#) ⇒ [Array.<Coordinate>](#)
  - [.intersectPointToLine\(p, line\)](#) ⇒ [Coordinate](#)
  - [.getPercentageDistanceFromPoint\(\\_coordinate\)](#) ⇒ Object
  - [.isContainsPoint\(\\_coordinate\)](#) ⇒ boolean
  - [.getPointFromPercentageDistance\(pXpY\)](#) ⇒ [Coordinate](#)
  - [.getParallelLine\(from, to, distance\)](#) ⇒ [Array.<Coordinate>](#)
  - [.getParallelPath\(line, distance\)](#)
  - [.reset\(\)](#)
- 
- [.Style](#) ≈ [HashMap](#)
    - [new OG.geometry.Style\(style\)](#)
    - [.map](#) : Object
    - [.put\(key, value\)](#)
    - [.get\(key\)](#) ⇒ Object
    - [.containsKey\(key\)](#) ⇒ Boolean
    - [.containsValue\(value\)](#) ⇒ Boolean
    - [.isEmpty\(\)](#) ⇒ Boolean
    - [.clear\(\)](#)
    - [.remove\(key\)](#)
    - [.keys\(\)](#) ⇒ [Array.<String>](#)

- [.values\(\)](#) ⇒ `Array.<Object>`
- [.size\(\)](#) ⇒ `Number`
- [.toString\(\)](#) ⇒ `String`

`geometry.BezierCurve` ⇐ [PolyLine](#)

Kind: static class of [geometry](#)

Extends: [PolyLine](#)

Requires: module:OG.geometry.Coordinate, module:OG.geometry.Envelope, module:OG.geometry.Geometry, module:OG.common.CurveUtil

Author: [Seungpil Park \(mailto:sppark@uengine.org\)](mailto:Seungpil.Park@sppark@uengine.org)

- [.BezierCurve](#) ⇐ [PolyLine](#)
  - [new OG.geometry.BezierCurve\(controlPoints\)](#)
  - [.controlPoints](#) : [Array.<Coordinate>](#)
  - [.vertices](#) : [Array.<Coordinate>](#)
  - [.TYPE](#) : `Number`
  - [.IS\\_CLOSED](#) : `Boolean`
  - [.style](#) : [Style](#)
  - [.boundary](#) : [Envelope](#)
  - [.getControlPoints\(\)](#) ⇒ [Array.<Coordinate>](#)
  - [.getVertices\(\)](#) ⇒ [Array.<Coordinate>](#)
  - [.move\(offsetX, offsetY\)](#) ⇒ [Geometry](#)

- [.resize\(upper, lower, left, right\)](#) ⇒ [Geometry](#)
- [.rotate\(angle, origin\)](#) ⇒ [Geometry](#)
- [.toString\(\)](#) ⇒ String
- [.angleBetweenPoints\(prev, next\)](#) ⇒ Number
- [.isRightAngleBetweenPoints\(prev, next\)](#) ⇒ Object
- [.angleBetweenThreePoints\(prev, next\)](#) ⇒ Number
- [.isEqual\(\\_geometry\)](#) ⇒ Boolean
- [.isContains\(\\_geometry\)](#) ⇒ Boolean
- [.isWithin\(\\_geometry\)](#) ⇒ Boolean
- [.getBoundary\(\)](#) ⇒ [Envelope](#)
- [.getCentroid\(\)](#) ⇒ [Coordinate](#)
- [.minDistance\(\\_coordinate\)](#) ⇒ Number
- [.distance\(\\_geometry\)](#) ⇒ Number
- [.getLength\(\)](#) ⇒ Number
- [.moveCentroid\(중심\)](#)
- [.resizeBox\(width, height\)](#) ⇒ [Geometry](#)
- [.fitToBoundary\(envelope\)](#) ⇒ [Geometry](#)
- [.convertCoordinate\(coordinate\)](#) ⇒ [Coordinate](#)
- [.distanceToLine\(p, line\)](#) ⇒ Number
- [.distanceLineToLine\(line1, line2\)](#) ⇒ Number
- [.intersectToLine\(line\)](#) ⇒ [Array.<Coordinate>](#)
- [.shortestIntersectToLine\(line\)](#) ⇒ [Array.<Coordinate>](#)
- [.intersectLineToLine\(line1, line2, extension\)](#) ⇒ [Coordinate](#)
- [.intersectCircleToLine\(center, radius, from, to\)](#) ⇒ [Array.<Coordinate>](#)
- [.intersectPointToLine\(p, line\)](#) ⇒ [Coordinate](#)
- [.getPercentageDistanceFromPoint\(\\_coordinate\)](#) ⇒ Object

- [.isContainsPoint\(\\_coordinate\)](#) ⇒ boolean
- [.getPointFromPercentageDistance\(pXpY\)](#) ⇒ [Coordinate](#)
- [.getParallelLine\(from, to, distance\)](#) ⇒ [Array.<Coordinate>](#)
- [.getParallelPath\(line, distance\)](#)
- [.reset\(\)](#)

```
new OG.geometry.BezierCurve(controlPoints)
```

Cubic Bezier Curve 공간 기하 객체(Spatial Geometry Object)

콘트롤팝인트1, 콘트롤팝인트2에 의해 시작좌표, 끝좌표를 지나는 곡선을 나타낸다.

Param	Type	Description
controlPoints	<a href="#">Array.&lt;Coordinate&gt;</a> [from, control_point1, control_point2, to]	

Example

```
var geom = new OG.geometry.BezierCurve([[200, 100], [100, 300], [-100, -100], [-200, 100]]);
```

```
bezierCurve.controlPoints : Array.<Coordinate>
```

Bezier Curve 콘트롤팝 좌표 Array

Kind: instance property of [BezierCurve](#)

```
bezierCurve.vertices : Array.<Coordinate>
```

Line Vertex 좌표 Array

Kind: instance property of [BezierCurve](#)

Overrides: [vertices](#)

bezierCurve.TYPE : Number

공간 기하 객체 타입

Kind: instance property of [BezierCurve](#)

Overrides: [TYPE](#)

bezierCurve.IS\_CLOSED : Boolean

닫힌 기하 객체 인지 여부

Kind: instance property of [BezierCurve](#)

bezierCurve.style : [Style](#)

스타일 속성

Kind: instance property of [BezierCurve](#)

Overrides: [style](#)

bezierCurve.boundary : [Envelope](#)

공간기하객체를 포함하는 사각형의 Boundary 영역

Kind: instance property of [BezierCurve](#)

Overrides: [boundary](#)

bezierCurve.getControlPoints() ⇒ [Array.<Coordinate>](#)

콘트롤 포인트 목록을 반환한다.

Kind: instance method of [BezierCurve](#)

Returns: [Array.<Coordinate>](#) - controlPoints Array

bezierCurve.getVertices() ⇒ [Array.<Coordinate>](#)

공간기하객체의 모든 꼭지점을 반환한다.

Kind: instance method of [BezierCurve](#)

Overrides: [getVertices](#)

Returns: [Array.<Coordinate>](#) - 꼭지점 좌표 Array

bezierCurve.move(offsetX, offsetY) ⇒ [Geometry](#)

가로, 세로 Offset 만큼 좌표를 이동한다.

Kind: instance method of [BezierCurve](#)

Overrides: [move](#)

Returns: [Geometry](#) - 이동된 공간 기하 객체

Param Type Description  
offsetX Number 가로 Offset  
offsetY Number 세로 Offset

`bezierCurve.resize(upper, lower, left, right) ⇒ Geometry`

상, 하, 좌, 우 외곽선을 이동하여 Envelope 을 리사이즈 한다.

Kind: instance method of [BezierCurve](#)

Overrides: [resize](#)

Returns: [Geometry](#) - 리사이즈된 공간 기하 객체

Param Type Description  
upper Number 상단 라인 이동 Offset(위 방향으로 +)  
lower Number 하단 라인 이동 Offset(아래 방향으로 +)  
left Number 좌측 라인 이동 Offset(좌측 방향으로 +)  
right Number 우측 라인 이동 Offset(우측 방향으로 +)

`bezierCurve.rotate(angle, origin) ⇒ Geometry`

기준 좌표를 기준으로 주어진 각도 만큼 회전한다.

Kind: instance method of [BezierCurve](#)

Overrides: [rotate](#)

Returns: [Geometry](#) - 회전된 공간 기하 객체

Param Type Description  
angle Number 회전 각도  
origin [Coordinate](#) 기준 좌표(default: 중심좌표)

```
bezierCurve.toString() => String
```

객체 프라퍼티 정보를 JSON 스트링으로 반환한다.

Kind: instance method of [BezierCurve](#)

Overrides: [toString](#)

Returns: String - 프라퍼티 정보

```
bezierCurve.angleBetweenPoints(prev, next) => Number
```

공간기하객체의 두 꼭지점 사이에 가상의 선을 그렸을때, 그 기울기를 구한다.

Kind: instance method of [BezierCurve](#)

Returns: Number - 기울기

Param Type Description

prev [Coordinate](#)꼭지점 1

next [Coordinate](#)꼭지점 2

```
bezierCurve.isRightAngleBetweenPoints(prev, next) => Object
```

공간기하객체의 두 꼭지점 사이의 기울기가 수평또는 수직인지 판별한다.

Kind: instance method of [BezierCurve](#)

Returns: Object - {flag : true or false, type: horizontal or vertical or none}

Param Type Description

prev [Coordinate](#)꼭지점 1

next [Coordinate](#)꼭지점 2

```
bezierCurve.angleBetweenThreePoints(prev, next) => Number
```

공간기하객체의 세 꼭지점 사이의 각도 중 작은 각도를 반환한다.

Kind: instance method of [BezierCurve](#)

Returns: Number - 기울기

Param Type Description

prev [Coordinate](#) 꼭지점 1

next [Coordinate](#) 꼭지점 2

```
bezierCurve.isEquals(_geometry) => Boolean
```

주어진 Geometry 객체와 같은지 비교한다.

Kind: instance method of [BezierCurve](#)

Returns: Boolean - true:같음, false:다름

Param Type Description

\_geometry [Geometry](#) Geometry 객체

```
bezierCurve.isContains(_geometry) => Boolean
```

주어진 공간기하객체를 포함하는지 비교한다.

Kind: instance method of [BezierCurve](#)

Returns: Boolean - 포함하면 true

Param	Type	Description
_geometry	<a href="#">Geometry</a>	Geometry 객체

`bezierCurve.isWithin(_geometry) ⇒ Boolean`

주어진 공간기하객체에 포함되는지 비교한다.

Kind: instance method of [BezierCurve](#)

Returns: Boolean - 포함되면 true

Param	Type	Description
_geometry	<a href="#">Geometry</a>	Geometry 객체

`bezierCurve.getBoundary() ⇒ Envelope`

공간기하객체를 포함하는 사각형의 Boundary 영역을 반환한다.

Kind: instance method of [BezierCurve](#)

Returns: [Envelope](#) - Envelope 영역

`bezierCurve.getCentroid() ⇒ Coordinate`

공간기하객체의 중심좌표를 반환한다.

Kind: instance method of [BezierCurve](#)

Returns: [Coordinate](#) - 중심좌표

`bezierCurve.minDistance(_coordinate) ⇒ Number`

주어진 좌표와의 최단거리를 반환한다.

Kind: instance method of [BezierCurve](#)

Returns: Number - 최단거리

Param	Type	Description
_coordinate	<a href="#">Coordinate</a>	좌표

```
bezierCurve.distance(_geometry) => Number
```

주어진 공간기하객체와의 중심점 간의 거리를 반환한다.

Kind: instance method of [BezierCurve](#)

Returns: Number - 거리

Param	Type	Description
_geometry	<a href="#">Geometry</a>	공간 기하 객체

```
bezierCurve.getLength() => Number
```

공간기하객체의 길이를 반환한다.

Kind: instance method of [BezierCurve](#)

Returns: Number - 길이

```
bezierCurve.moveCentroid(중심)
```

주어진 중심좌표로 공간기하객체를 이동한다.

Kind: instance method of [BezierCurve](#)

Param Type Description  
중심 [Coordinate](#) 좌표

`bezierCurve.resizeBox(width, height) ⇒ Geometry`

중심좌표는 고정한 채 Bounding Box 의 width, height 를 리사이즈 한다.

Kind: instance method of [BezierCurve](#)

Returns: [Geometry](#) - 리사이즈된 공간 기하 객체

Param Type Description  
width Number 너비  
height Number 높이

`bezierCurve.fitToBoundary(envelope) ⇒ Geometry`

주어진 Boundary 영역 안으로 공간 기하 객체를 적용한다.(이동 & 리사이즈)

Kind: instance method of [BezierCurve](#)

Returns: [Geometry](#) - 적용된 공간 기하 객체

Param Type Description  
envelope [Envelope](#) Envelope 영역

`bezierCurve.convertCoordinate(coordinate) ⇒ Coordinate`

파라미터가 [x, y] 형식의 좌표 Array 이면 OG.geometry.Coordinate 인스턴스를 new 하여 반환한다.

Kind: instance method of [BezierCurve](#)

Param	Type	Description
coordinate <code>&lt;Number&gt;</code>	<code>Coordinate</code>   Array. <code>&lt;Number&gt;</code>	[x, y] 형식의 좌표 Array 또는 OG.geometry.Coordinate 인스턴스

`bezierCurve.distanceToLine(p, line) ⇒ Number`

포인트 P로부터 라인 AB의 거리를 계산한다.

Note: NON-ROBUST!

Kind: instance method of [BezierCurve](#)

Returns: Number - 거리

Param	Type	Description
p	<code>Coordinate</code>   Array. <code>&lt;Number&gt;</code>	기준좌표
line	<code>Array.&lt;Coordinate&gt;</code>	라인 시작좌표, 끝좌표 Array

`bezierCurve.distanceLineToLine(line1, line2) ⇒ Number`

라인1로부터 라인2의 거리를 계산한다.

Note: NON-ROBUST!

Kind: instance method of [BezierCurve](#)

Returns: Number - 거리

Param	Type	Description
line1	<code>Array.&lt;Coordinate&gt;</code>	line1 라인 시작좌표, 끝좌표 Array
line2	<code>Array.&lt;Coordinate&gt;</code>	line2 라인 시작좌표, 끝좌표 Array

```
bezierCurve.intersectToLine(line) ⇒ Array.<Coordinate>
```

기하도형이 주어진 라인과 교차하는 좌표들을 반환한다.

Kind: instance method of [BezierCurve](#)

Param	Type	Description
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 시작좌표, 끝좌표 Array

```
bezierCurve.shortestIntersectToLine(line) ⇒ Array.<Coordinate>
```

기하도형이 주어진 라인과 교차하는 좌표중 시작좌표에 가장 가까운 좌표를 반환한다.

Kind: instance method of [BezierCurve](#)

Param	Type	Description
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 시작좌표, 끝좌표 Array

```
bezierCurve.intersectLineToLine(line1, line2, extension) ⇒ Coordinate
```

라인1 과 라인2 의 교차점을 계산한다.

Kind: instance method of [BezierCurve](#)

Returns: [Coordinate](#) - 교차점

Param	Type	Description
line1	<a href="#">Array.&lt;Coordinate&gt;</a>	line1 라인 시작좌표, 끝좌표 Array
line2	<a href="#">Array.&lt;Coordinate&gt;</a>	line2 라인 시작좌표, 끝좌표 Array
extension	boolean	라인을 연장하여 교차점을 계산하는 여부

```
bezierCurve.intersectCircleToLine(center, radius, from, to) => Array.<Coordinate>
```

주어진 원과 주어진 라인의 교차점을 계산한다.

Kind: instance method of [BezierCurve](#)

Returns: [Array.<Coordinate>](#) - 교차점

Param	Type	Description
center	<a href="#">Coordinate</a>	중심점
radius	Number	반경
from	<a href="#">Coordinate</a>	line 라인 시작좌표
to	<a href="#">Coordinate</a>	line 라인 끝좌표

```
bezierCurve.intersectPointToLine(p, line) => Coordinate
```

포인트 P로부터 라인 AB 까지 수직인 가상선을 생각할때, 그 교차점을 계산한다.

Note: NON-ROBUST!

Kind: instance method of [BezierCurve](#)

Returns: [Coordinate](#) - 교차점

Param	Type	Description
p	<a href="#">Coordinate</a>   <a href="#">Array.&lt;Number&gt;</a>	기준좌표
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 시작좌표, 끝좌표 Array

```
bezierCurve.getPercentageDistanceFromPoint(_coordinate) => Object
```

주어진 좌표에 대해 공간기하객체 바운더리 대비 가로, 세로 위치 퍼센테이지 비율을 구한다.

Kind: instance method of [BezierCurve](#)

Returns: Object - {px , py}

Param	Type	Description
_coordinate	<a href="#">Coordinate</a>	좌표

bezierCurve.isContainsPoint(\_coordinate) ⇒ boolean

공간기하객체가 주어진 좌표를 포함하는지를 반환한다.

Kind: instance method of [BezierCurve](#)

Returns: boolean - true, false

Param	Type	Description
_coordinate	<a href="#">Coordinate</a>	좌표

bezierCurve.getPointFromPercentageDistance(pXpY) ⇒ [Coordinate](#)

공간기하객체에 대한 퍼센테이지 좌표의 실제 좌표를 구한다.

Kind: instance method of [BezierCurve](#)

Returns: [Coordinate](#) - 실 좌표

Param	Type	Description
pXpY	Array	퍼센테이지 좌표

bezierCurve.getParallelLine(from, to, distance) ⇒ [Array.<Coordinate>](#)

주어진 선분과 일정 거리에 있는 평행한 선분을 반환한다.

Kind: instance method of [BezierCurve](#)

Returns: [Array.<Coordinate>](#) - 평행선 시작좌표, 끝좌표 Array

Param	Type	Description
from	<a href="#">Coordinate</a>	라인 시작좌표
to	<a href="#">Coordinate</a>	라인 끝좌표
distance		

```
bezierCurve.getParallelPath(line, distance)
```

주어진 라인과 일정 거리에 있는 평행한 라인을 반환한다.

Kind: instance method of [BezierCurve](#)

Param	Type	Description
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 좌표 Array
distance		

```
bezierCurve.reset()
```

저장된 boundary 를 클리어하여 새로 계산하도록 한다.

Kind: instance method of [BezierCurve](#)

Overrides: [reset](#)

```
geometry.Circle <= Ellipse
```

Kind: static class of [geometry](#)

Extends: [Ellipse](#)

Requires: module:OG.geometry.Coordinate, module:OG.geometry.Envelope,  
module:OG.geometry.Geometry

Author: [Seungpil Park \(mailto:spspark@uengine.org\)](mailto:Seungpil.Park@sengine.org)

- [.Circle](#)  $\Leftarrow$  [Ellipse](#)
  - [new OG.geometry.Circle\(center, radius\)](#)
  - [.vertices : Array.<Coordinate>](#)
  - [.TYPE : Number](#)
  - [.IS\\_CLOSED : Boolean](#)
  - [.style : Style](#)
  - [.boundary : Envelope](#)
  - [.getVertices\(\)  \$\Rightarrow\$  Array.<Coordinate>](#)
  - [.getControlPoints\(\)  \$\Rightarrow\$  Array.<Coordinate>](#)
  - [.getLength\(\)  \$\Rightarrow\$  Number](#)
  - [.toString\(\)  \$\Rightarrow\$  String](#)
  - [.move\(offsetX, offsetY\)  \$\Rightarrow\$  Geometry](#)
  - [.resize\(upper, lower, left, right\)  \$\Rightarrow\$  Geometry](#)
  - [.rotate\(angle, origin\)  \$\Rightarrow\$  Geometry](#)
  - [.angleBetweenPoints\(prev, next\)  \$\Rightarrow\$  Number](#)
  - [.isRightAngleBetweenPoints\(prev, next\)  \$\Rightarrow\$  Object](#)
  - [.angleBetweenThreePoints\(prev, next\)  \$\Rightarrow\$  Number](#)
  - [.isEqual\(geometry\)  \$\Rightarrow\$  Boolean](#)
  - [.isContains\(geometry\)  \$\Rightarrow\$  Boolean](#)
  - [.isWithin\(geometry\)  \$\Rightarrow\$  Boolean](#)
  - [.getBoundary\(\)  \$\Rightarrow\$  Envelope](#)

- [.getCentroid\(\)](#) ⇒ [Coordinate](#)
- [.minDistance\(\\_coordinate\)](#) ⇒ Number
- [.distance\(\\_geometry\)](#) ⇒ Number
- [.moveCentroid\(중심\)](#)
- [.resizeBox\(width, height\)](#) ⇒ [Geometry](#)
- [.fitToBoundary\(envelope\)](#) ⇒ [Geometry](#)
- [.convertCoordinate\(coordinate\)](#) ⇒ [Coordinate](#)
- [.distanceToLine\(p, line\)](#) ⇒ Number
- [.distanceLineToLine\(line1, line2\)](#) ⇒ Number
- [.intersectToLine\(line\)](#) ⇒ [Array.<Coordinate>](#)
- [.shortestIntersectToLine\(line\)](#) ⇒ [Array.<Coordinate>](#)
- [.intersectLineToLine\(line1, line2, extension\)](#) ⇒ [Coordinate](#)
- [.intersectCircleToLine\(center, radius, from, to\)](#) ⇒ [Array.<Coordinate>](#)
- [.intersectPointToLine\(p, line\)](#) ⇒ [Coordinate](#)
- [.getPercentageDistanceFromPoint\(\\_coordinate\)](#) ⇒ Object
- [.isContainsPoint\(\\_coordinate\)](#) ⇒ boolean
- [.getPointFromPercentageDistance\(pXpY\)](#) ⇒ [Coordinate](#)
- [.getParallelLine\(from, to, distance\)](#) ⇒ [Array.<Coordinate>](#)
- [.getParallelPath\(line, distance\)](#)
- [.reset\(\)](#)

new OG.geometry.Circle(center, radius)

Circle 공간 기하 객체(Spatial Geometry Object)

Param	Type	Description
center	<a href="#">Coordinate</a>	Circle 중심 좌표
radius	Number	radius 반경

## Example

```
var geom = new OG.geometry.Circle([10, 10], 5);
```

circle.vertices : [Array.<Coordinate>](#)

## Line Vertex 좌표 Array

Kind: instance property of [Circle](#)

circle.TYPE : Number

공간 기하 객체 타입

Kind: instance property of [Circle](#)

Overrides: [TYPE](#)

circle.IS\_CLOSED : Boolean

닫힌 기하 객체 인지 여부

Kind: instance property of [Circle](#)

circle.style : [Style](#)

스타일 속성

Kind: instance property of [Circle](#)

Overrides: [style](#)

circle.boundary : [Envelope](#)

공간기하객체를 포함하는 사각형의 Boundary 영역

Kind: instance property of [Circle](#)

circle.getVertices() => [Array.<Coordinate>](#)

공간기하객체의 모든 꼭지점을 반환한다.

Kind: instance method of [Circle](#)

Returns: [Array.<Coordinate>](#) - 꼭지점 좌표 Array

circle.getControlPoints() => [Array.<Coordinate>](#)

콘트롤 포인트 목록을 반환한다.

Kind: instance method of [Circle](#)

Returns: [Array.<Coordinate>](#) - controlPoints Array

```
circle.getLength() => Number
```

공간기하 객체의 길이를 반환한다.

Kind: instance method of [Circle](#)

Overrides: [getLength](#)

Returns: Number - 길이

```
circle.toString() => String
```

객체 프라퍼티 정보를 JSON 스트링으로 반환한다.

Kind: instance method of [Circle](#)

Overrides: [toString](#)

Returns: String - 프라퍼티 정보

```
circle.move(offsetX, offsetY) => Geometry
```

가로, 세로 Offset 만큼 좌표를 이동한다.

Kind: instance method of [Circle](#)

Returns: [Geometry](#) - 이동된 공간 기하 객체

Param	Type	Description
offsetX	Number	가로 Offset
offsetY	Number	세로 Offset

```
circle.resize(upper, lower, left, right) => Geometry
```

상, 하, 좌, 우 외곽선을 이동하여 Envelope 을 리사이즈 한다.

Kind: instance method of [Circle](#)

Returns: [Geometry](#) - 리사이즈된 공간 기하 객체

Param	Type	Description
upper	Number	상단 라인 이동 Offset(위 방향으로 +)
lower	Number	하단 라인 이동 Offset(아래 방향으로 +)
left	Number	좌측 라인 이동 Offset(좌측 방향으로 +)
right	Number	우측 라인 이동 Offset(우측 방향으로 +)

circle.rotate(angle, origin) ⇒ [Geometry](#)

기준 좌표를 기준으로 주어진 각도 만큼 회전한다.

Kind: instance method of [Circle](#)

Returns: [Geometry](#) - 회전된 공간 기하 객체

Param	Type	Description
angle	Number	회전 각도
origin	<a href="#">Coordinate</a>	기준 좌표(default:중심좌표)

circle.angleBetweenPoints(prev, next) ⇒ Number

공간기하객체의 두 꼭지점 사이에 가상의 선을 그렸을때, 그 기울기를 구한다.

Kind: instance method of [Circle](#)

Returns: Number - 기울기

Param	Type	Description
prev	<a href="#">Coordinate</a>	꼭지점 1

next [Coordinate](#) 꼭지점 2

```
circle.isRightAngleBetweenPoints(prev, next) => Object
```

공간기하객체의 두 꼭지점 사이의 기울기가 수평또는 수직인지 판별한다.

Kind: instance method of [Circle](#)

Returns: Object - {flag : true or false, type: horizontal or vertical or none}

Param Type Description

prev [Coordinate](#) 꼭지점 1

next [Coordinate](#) 꼭지점 2

```
circle.angleBetweenThreePoints(prev, next) => Number
```

공간기하객체의 세 꼭지점 사이의 각도 중 작은 각도를 반환한다.

Kind: instance method of [Circle](#)

Returns: Number - 기울기

Param Type Description

prev [Coordinate](#) 꼭지점 1

next [Coordinate](#) 꼭지점 2

```
circle.isEquals(_geometry) => Boolean
```

주어진 Geometry 객체와 같은지 비교한다.

Kind: instance method of [Circle](#)

Returns: Boolean - true:같음, false:다름

Param	Type	Description
_geometry	<a href="#">Geometry</a>	Geometry 객체

```
circle.isContains(_geometry) ⇒ Boolean
```

주어진 공간기하객체를 포함하는지 비교한다.

Kind: instance method of [Circle](#)

Returns: Boolean - 포함하면 true

Param	Type	Description
_geometry	<a href="#">Geometry</a>	Geometry 객체

```
circle.isWithin(_geometry) ⇒ Boolean
```

주어진 공간기하객체에 포함되는지 비교한다.

Kind: instance method of [Circle](#)

Returns: Boolean - 포함되면 true

Param	Type	Description
_geometry	<a href="#">Geometry</a>	Geometry 객체

```
circle.getBoundary() ⇒ Envelope
```

공간기하객체를 포함하는 사각형의 Boundary 영역을 반환한다.

Kind: instance method of [Circle](#)

Returns: [Envelope](#) - Envelope 영역

circle.getCentroid() => [Coordinate](#)

공간기하객체의 중심좌표를 반환한다.

Kind: instance method of [Circle](#)

Returns: [Coordinate](#) - 중심좌표

circle.minDistance(\_coordinate) => Number

주어진 좌표와의 최단거리를 반환한다.

Kind: instance method of [Circle](#)

Returns: Number - 최단거리

Param	Type	Description
_coordinate	<a href="#">Coordinate</a>	좌표

circle.distance(\_geometry) => Number

주어진 공간기하객체와의 중심점 간의 거리를 반환한다.

Kind: instance method of [Circle](#)

Returns: Number - 거리

Param	Type	Description
_geometry	<a href="#">Geometry</a>	공간 기하 객체

circle.moveCentroid([중심](#))

주어진 중심좌표로 공간기하객체를 이동한다.

Kind: instance method of [Circle](#)

Param	Type	Description
중심	<a href="#">Coordinate</a>	좌표

circle.resizeBox(width, height) ⇒ [Geometry](#)

중심좌표는 고정한 채 Bounding Box 의 width, height 를 리사이즈 한다.

Kind: instance method of [Circle](#)

Returns: [Geometry](#) - 리사이즈된 공간 기하 객체

Param	Type	Description
width	Number	너비
height	Number	높이

circle.fitToBoundary(envelope) ⇒ [Geometry](#)

주어진 Boundary 영역 안으로 공간 기하 객체를 적용한다.(이동 & 리사이즈)

Kind: instance method of [Circle](#)

Returns: [Geometry](#) - 적용된 공간 기하 객체

Param	Type	Description
envelope	<a href="#">Envelope</a>	Envelope 영역

circle.convertCoordinate(coordinate) ⇒ [Coordinate](#)

파라미터가 [x, y] 형식의 좌표 Array 이면 OG.geometry.Coordinate 인스턴스를 new 하여 반환한다.

Kind: instance method of [Circle](#)

Param	Type	Description
coordinate	<a href="#">Coordinate</a>   Array. <Number>	[x, y] 형식의 좌표 Array 또는 OG.geometry.Coordinate 인스턴스

circle.distanceToLine(p, line) ⇒ Number

포인트 P로부터 라인 AB의 거리를 계산한다.

Note: NON-ROBUST!

Kind: instance method of [Circle](#)

Returns: Number - 거리

Param	Type	Description
p	<a href="#">Coordinate</a>   Array.<Number>	기준좌표
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 시작좌표, 끝좌표 Array

circle.distanceLineToLine(line1, line2) ⇒ Number

라인1로부터 라인2의 거리를 계산한다.

Note: NON-ROBUST!

Kind: instance method of [Circle](#)

Returns: Number - 거리

Param Type Description

line1 [Array.<Coordinate>](#) 라인 시작좌표, 끝좌표 Array

line2 [Array.<Coordinate>](#) 라인 시작좌표, 끝좌표 Array

`circle.intersectToLine(line) ⇒ Array.<Coordinate>`

기하도형이 주어진 라인과 교차하는 좌표들을 반환한다.

Kind: instance method of [Circle](#)

Param Type Description

line [Array.<Coordinate>](#) 라인 시작좌표, 끝좌표 Array

`circle.shortestIntersectToLine(line) ⇒ Array.<Coordinate>`

기하도형이 주어진 라인과 교차하는 좌표중 시작좌표에 가장 가까운 좌표를 반환한다.

Kind: instance method of [Circle](#)

Param Type Description

line [Array.<Coordinate>](#) 라인 시작좌표, 끝좌표 Array

`circle.intersectLineToLine(line1, line2, extension) ⇒ Coordinate`

라인1 과 라인2 의 교차점을 계산한다.

Kind: instance method of [Circle](#)

Returns: [Coordinate](#) - 교차점

Param	Type	Description
line1	<a href="#">Array.&lt;Coordinate&gt;</a>	line1 라인 시작좌표, 끝좌표 Array
line2	<a href="#">Array.&lt;Coordinate&gt;</a>	line2 라인 시작좌표, 끝좌표 Array
extension	boolean	라인을 연장하여 교차점을 계산하는 여부

`circle.intersectCircleToLine(center, radius, from, to) ⇒ Array.<Coordinate>`

주어진 원과 주어진 라인의 교차점을 계산한다.

Kind: instance method of [Circle](#)

Returns: [Array.<Coordinate>](#) - 교차점

Param	Type	Description
center	<a href="#">Coordinate</a>	중심점
radius	Number	반경
from	<a href="#">Coordinate</a>	라인 시작좌표
to	<a href="#">Coordinate</a>	라인 끝좌표

`circle.intersectPointToLine(p, line) ⇒ Coordinate`

포인트 P로부터 라인 AB 까지 수직인 가상선을 생각할때, 그 교차점을 계산한다.

Note: NON-ROBUST!

Kind: instance method of [Circle](#)

Returns: [Coordinate](#) - 교차점

Param	Type	Description
p	<a href="#">Coordinate</a>   <a href="#">Array.&lt;Number&gt;</a>	기준좌표
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 시작좌표, 끝좌표 Array

```
circle.getPercentageDistanceFromPoint(_coordinate) => Object
```

주어진 좌표에 대해 공간기하객체 바운더리 대비 가로, 세로 위치 퍼센테이지 비율을 구한다.

Kind: instance method of [Circle](#)

Returns: Object - {px , py}

Param	Type	Description
_coordinate	<a href="#">Coordinate</a>	좌표

```
circle.isContainsPoint(_coordinate) => boolean
```

공간기하객체가 주어진 좌표를 포함하는지를 반환한다.

Kind: instance method of [Circle](#)

Returns: boolean - true, false

Param	Type	Description
_coordinate	<a href="#">Coordinate</a>	좌표

```
circle.getPointFromPercentageDistance(pXpY) => Coordinate
```

공간기하객체에 대한 퍼센테이지 좌표의 실제 좌표를 구한다.

Kind: instance method of [Circle](#)

Returns: [Coordinate](#) - 실 좌표

Param	Type	Description
-------	------	-------------

pXpY Array 퍼센테이지 좌표

circle.getParallelLine(from, to, distance) ⇒ [Array.<Coordinate>](#)

주어진 선분과 일정 거리에 있는 평행한 선분을 반환한다.

Kind: instance method of [Circle](#)

Returns: [Array.<Coordinate>](#) - 평행선 시작좌표, 끝좌표 Array

Param	Type	Description
from	<a href="#">Coordinate</a>	라인 시작좌표
to	<a href="#">Coordinate</a>	라인 끝좌표
distance		

circle.getParallelPath(line, distance)

주어진 라인과 일정 거리에 있는 평행한 라인을 반환한다.

Kind: instance method of [Circle](#)

Param	Type	Description
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 좌표 Array
distance		

circle.reset()

저장된 boundary 를 클리어하여 새로 계산하도록 한다.

Kind: instance method of [Circle](#)

geometry.Coordinate

Kind: static class of [geometry](#)

Author: [Seungpil Park \(mailto:sppark@uengine.org\)](mailto:Seungpil.Park@sppark.org)

- [.Coordinate](#)

- [new OG.geometry.Coordinate\(x, y\)](#)
- [.x](#) : Number
- [.y](#) : Number
- [.distance\(coordinate\)](#) ⇒ Number
- [.move\(offsetX, offsetY\)](#) ⇒ [Coordinate](#)
- [.rotate\(angle, origin\)](#) ⇒ [Coordinate](#)
- [.isEquals\(coordinate\)](#) ⇒ Boolean
- [.toString\(\)](#) ⇒ String

`new OG.geometry.Coordinate(x, y)`

2차원 좌표계에서의 좌표값

Param Type Description

x Number x좌표

y Number y좌표

Example

```
var coordinate1 = new OG.geometry.Coordinate(10, 10);
or
var coordinate2 = new OG.geometry.Coordinate([20, 20]);
```

coordinate.x : Number

x좌표

Kind: instance property of [Coordinate](#)

coordinate.y : Number

y좌표

Kind: instance property of [Coordinate](#)

coordinate.distance(coordinate) ⇒ Number

주어진 좌표와의 거리를 계산한다.

Kind: instance method of [Coordinate](#)

Returns: Number - 좌표간의 거리값

Param	Type	Description
coordinate	<a href="#">Coordinate</a>   Array.<Number>	좌표값

Example

```
coordinate.distance([10, 10]);  
or  
coordinate.distance(new OG.Coordinate(10, 10));
```

coordinate.move(offsetX, offsetY) ⇒ [Coordinate](#)

가로, 세로 Offset 만큼 좌표를 이동한다.

Kind: instance method of [Coordinate](#)

Returns: [Coordinate](#) - 이동된 좌표

Param	Type	Description
offsetX	Number	가로 Offset
offsetY	Number	세로 Offset

coordinate.rotate(angle, origin) ⇒ [Coordinate](#)

기준 좌표를 기준으로 주어진 각도 만큼 회전한다.

Kind: instance method of [Coordinate](#)

Returns: [Coordinate](#) - 회전된 좌표

Param	Type	Description
angle	Number	회전 각도
origin	<a href="#">Coordinate</a>   Array.<Number>	기준 좌표

Example

```
coordinate.rotate(90, [10, 10]);  
or  
coordinate.rotate(90, new OG.Coordinate(10, 10));
```

```
coordinate.equals(coordinate) => Boolean
```

주어진 좌표값과 같은지 비교한다.

Kind: instance method of [Coordinate](#)

Returns: Boolean - true:같음, false:다름

Param	Type	Description
coordinate	<a href="#">Coordinate</a>   Array.<Number>	좌표값

Example

```
coordinate.equals([10, 10]);  
or  
coordinate.equals(new OG.Coordinate(10, 10));
```

```
coordinate.toString() => String
```

객체 프라퍼티 정보를 JSON 스트링으로 반환한다.

Kind: instance method of [Coordinate](#)

Returns: String - 프라퍼티 정보

```
geometry.Curve <- PolyLine
```

Kind: static class of [geometry](#)

Extends: [PolyLine](#)

Requires: module:OG.geometry.Coordinate, module:OG.geometry.Envelope,

module:OG.geometry.Geometry, module:OG.common.CurveUtil

Author: [Seungpil Park \(mailto:sspark@uengine.org\)](mailto:sspark@uengine.org)

- [.Curve](#) ⇐ [PolyLine](#)
  - [new OG.geometry.Curve\(controlPoints\)](#)
  - [.vertices : Array.<Coordinate>](#)
  - [.TYPE : Number](#)
  - [.IS\\_CLOSED : Boolean](#)
  - [.style : Style](#)
  - [.boundary : Envelope](#)
  - [.getControlPoints\(\) ⇒ Array.<Coordinate>](#)
  - [.getVertices\(\) ⇒ Array.<Coordinate>](#)
  - [.move\(offsetX, offsetY\) ⇒ Geometry](#)
  - [.resize\(upper, lower, left, right\) ⇒ Geometry](#)
  - [.rotate\(angle, origin\) ⇒ Geometry](#)
  - [.toString\(\) ⇒ String](#)
  - [.angleBetweenPoints\(prev, next\) ⇒ Number](#)
  - [.isRightAngleBetweenPoints\(prev, next\) ⇒ Object](#)
  - [.angleBetweenThreePoints\(prev, next\) ⇒ Number](#)
  - [.isEqual\(\\_geometry\) ⇒ Boolean](#)
  - [.contains\(\\_geometry\) ⇒ Boolean](#)
  - [.within\(\\_geometry\) ⇒ Boolean](#)
  - [.getBoundary\(\) ⇒ Envelope](#)
  - [.getCentroid\(\) ⇒ Coordinate](#)
  - [.minDistance\(\\_coordinate\) ⇒ Number](#)

- [.distance\(geometry\)](#) ⇒ Number
- [.getLength\(\)](#) ⇒ Number
- [.moveCentroid\(중심\)](#)
- [.resizeBox\(width, height\)](#) ⇒ [Geometry](#)
- [.fitToBoundary\(envelope\)](#) ⇒ [Geometry](#)
- [.convertCoordinate\(coordinate\)](#) ⇒ [Coordinate](#)
- [.distanceToLine\(p, line\)](#) ⇒ Number
- [.distanceLineToLine\(line1, line2\)](#) ⇒ Number
- [.intersectToLine\(line\)](#) ⇒ [Array.<Coordinate>](#)
- [.shortestIntersectToLine\(line\)](#) ⇒ [Array.<Coordinate>](#)
- [.intersectLineToLine\(line1, line2, extension\)](#) ⇒ [Coordinate](#)
- [.intersectCircleToLine\(center, radius, from, to\)](#) ⇒ [Array.<Coordinate>](#)
- [.intersectPointToLine\(p, line\)](#) ⇒ [Coordinate](#)
- [.getPercentageDistanceFromPoint\(\\_coordinate\)](#) ⇒ Object
- [.isContainsPoint\(\\_coordinate\)](#) ⇒ boolean
- [.getPointFromPercentageDistance\(pXpY\)](#) ⇒ [Coordinate](#)
- [.getParallelLine\(from, to, distance\)](#) ⇒ [Array.<Coordinate>](#)
- [.getParallelPath\(line, distance\)](#)
- [.reset\(\)](#)

new OG.geometry.Curve(controlPoints)

Catmull-Rom Spline Curve 공간 기하 객체(Spatial Geometry Object)

모든 콘트롤포인트를 지나는 곡선을 나타낸다.

Param	Type	Description
controlPoints	<a href="#">Array.&lt;Coordinate&gt;</a>	Curve Vertex 좌표 Array

## Example

```
var geom = new OG.geometry.Curve([[200, 100], [100, 300], [-100, -100], [-200, 100]]);
```

curve.vertices : [Array.<Coordinate>](#)

Line Vertex 좌표 Array

Kind: instance property of [Curve](#)

Overrides: [vertices](#)

curve.TYPE : Number

공간 기하 객체 타입

Kind: instance property of [Curve](#)

Overrides: [TYPE](#)

curve.IS\_CLOSED : Boolean

닫힌 기하 객체 인지 여부

Kind: instance property of [Curve](#)

curve.style : [Style](#)

스타일 속성

Kind: instance property of [Curve](#)

Overrides: [style](#)

curve.boundary : [Envelope](#)

공간기하객체를 포함하는 사각형의 Boundary 영역

Kind: instance property of [Curve](#)

curve.getControlPoints() => [Array.<Coordinate>](#)

콘트롤 포인트 목록을 반환한다.

Kind: instance method of [Curve](#)

Returns: [Array.<Coordinate>](#) - controlPoints Array

curve.getVertices() => [Array.<Coordinate>](#)

공간기하객체의 모든 꼭지점을 반환한다.

Kind: instance method of [Curve](#)

Overrides: [getVertices](#)

Returns: [Array.<Coordinate>](#) - 꼭지점 좌표 Array

curve.move(offsetX, offsetY) ⇒ [Geometry](#)

가로, 세로 Offset 만큼 좌표를 이동한다.

Kind: instance method of [Curve](#)

Returns: [Geometry](#) - 이동된 공간 기하 객체

Param	Type	Description
offsetX	Number	가로 Offset
offsetY	Number	세로 Offset

curve.resize(upper, lower, left, right) ⇒ [Geometry](#)

상, 하, 좌, 우 외곽선을 이동하여 Envelope 을 리사이즈 한다.

Kind: instance method of [Curve](#)

Returns: [Geometry](#) - 리사이즈된 공간 기하 객체

Param	Type	Description
upper	Number	상단 라인 이동 Offset(위 방향으로 +)
lower	Number	하단 라인 이동 Offset(아래 방향으로 +)
left	Number	좌측 라인 이동 Offset(좌측 방향으로 +)
right	Number	우측 라인 이동 Offset(우측 방향으로 +)

curve.rotate(angle, origin) ⇒ [Geometry](#)

기준 좌표를 기준으로 주어진 각도 만큼 회전한다.

Kind: instance method of [Curve](#)

Returns: [Geometry](#) - 회전된 공간 기하 객체

Param	Type	Description
angle	Number	회전 각도
origin	<a href="#">Coordinate</a>	기준 좌표(default: 중심좌표)

curve.toString() ⇒ String

객체 프라퍼티 정보를 JSON 스트링으로 반환한다.

Kind: instance method of [Curve](#)

Overrides: [toString](#)

Returns: String - 프라퍼티 정보

curve.angleBetweenPoints(prev, next) ⇒ Number

공간기하객체의 두 꼭지점 사이에 가상의 선을 그렸을때, 그 기울기를 구한다.

Kind: instance method of [Curve](#)

Returns: Number - 기울기

Param	Type	Description
prev	<a href="#">Coordinate</a>	꼭지점 1
next	<a href="#">Coordinate</a>	꼭지점 2

curve.isRightAngleBetweenPoints(prev, next) ⇒ Object

공간기하객체의 두 꼭지점 사이의 기울기가 수평또는 수직인지 판별한다.

Kind: instance method of [Curve](#)

Returns: Object - {flag : true or false, type: horizontal or vertical or none}

Param Type Description

prev [Coordinate](#) 꼭지점 1

next [Coordinate](#) 꼭지점 2

curve.angleBetweenThreePoints(prev, next) ⇒ Number

공간기하객체의 세 꼭지점 사이의 각도 중 작은 각도를 반환한다.

Kind: instance method of [Curve](#)

Returns: Number - 기울기

Param Type Description

prev [Coordinate](#) 꼭지점 1

next [Coordinate](#) 꼭지점 2

curve.equals(\_geometry) ⇒ Boolean

주어진 Geometry 객체와 같은지 비교한다.

Kind: instance method of [Curve](#)

Returns: Boolean - true:같음, false:다름

Param Type Description

\_geometry [Geometry](#) Geometry 객체

curve.contains(\_geometry) ⇒ Boolean

주어진 공간기하 객체를 포함하는지 비교한다.

Kind: instance method of [Curve](#)

Returns: Boolean - 포함하면 true

Param	Type	Description
_geometry	<a href="#">Geometry</a>	Geometry 객체

curve.isWithin(\_geometry) ⇒ Boolean

주어진 공간기하 객체에 포함되는지 비교한다.

Kind: instance method of [Curve](#)

Returns: Boolean - 포함되면 true

Param	Type	Description
_geometry	<a href="#">Geometry</a>	Geometry 객체

curve.getBoundary() ⇒ [Envelope](#)

공간기하 객체를 포함하는 사각형의 Boundary 영역을 반환한다.

Kind: instance method of [Curve](#)

Returns: [Envelope](#) - Envelope 영역

curve.getCentroid() ⇒ [Coordinate](#)

공간기하 객체의 중심좌표를 반환한다.

Kind: instance method of [Curve](#)

Returns: [Coordinate](#) - 중심좌표

curve.minDistance(\_coordinate) => Number

주어진 좌표와의 최단거리를 반환한다.

Kind: instance method of [Curve](#)

Returns: Number - 최단거리

Param	Type	Description
_coordinate	<a href="#">Coordinate</a>	좌표

curve.distance(\_geometry) => Number

주어진 공간기하 객체와의 중심점 간의 거리를 반환한다.

Kind: instance method of [Curve](#)

Returns: Number - 거리

Param	Type	Description
_geometry	<a href="#">Geometry</a>	공간 기하 객체

curve.getLength() => Number

공간기하 객체의 길이를 반환한다.

Kind: instance method of [Curve](#)

Returns: Number - 길이

curve.moveToCentroid(중심)

주어진 중심좌표로 공간기하 객체를 이동한다.

Kind: instance method of [Curve](#)

Param Type Description

중심 [Coordinate](#) 좌표

curve.resizeBox(width, height) ⇒ [Geometry](#)

중심좌표는 고정한 채 Bounding Box 의 width, height 를 리사이즈 한다.

Kind: instance method of [Curve](#)

Returns: [Geometry](#) - 리사이즈된 공간 기하 객체

Param Type Description

width Number 너비

height Number 높이

curve.fitToBoundary(envelope) ⇒ [Geometry](#)

주어진 Boundary 영역 안으로 공간 기하 객체를 적용한다. (이동 & 리사이즈)

Kind: instance method of [Curve](#)

Returns: [Geometry](#) - 적용된 공간 기하 객체

Param	Type	Description
envelope	<a href="#">Envelope</a>	Envelope 영역

curve.convertCoordinate(coordinate) ⇒ [Coordinate](#)

파라미터가 [x, y] 형식의 좌표 Array 이면 OG.geometry.Coordinate 인스턴스를 new 하여 반환한다.

Kind: instance method of [Curve](#)

Param	Type	Description
coordinate	<a href="#">Coordinate</a>   Array. <Number>	[x, y] 형식의 좌표 Array 또는 OG.geometry.Coordinate 인스턴스

curve.distanceToLine(p, line) ⇒ Number

포인트 P로부터 라인 AB의 거리를 계산한다.

Note: NON-ROBUST!

Kind: instance method of [Curve](#)

Returns: Number - 거리

Param	Type	Description
p	<a href="#">Coordinate</a>   Array.<Number>	기준좌표
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 시작좌표, 끝좌표 Array

curve.distanceLineToLine(line1, line2) ⇒ Number

라인1로부터 라인2의 거리를 계산한다.

Note: NON-ROBUST!

Kind: instance method of [Curve](#)

Returns: Number - 거리

Param Type Description

line1 [Array.<Coordinate>](#) line1 라인 시작좌표, 끝좌표 Array

line2 [Array.<Coordinate>](#) line2 라인 시작좌표, 끝좌표 Array

curve.intersectToLine(line) ⇒ [Array.<Coordinate>](#)

기하도형이 주어진 라인과 교차하는 좌표들을 반환한다.

Kind: instance method of [Curve](#)

Param Type Description

line [Array.<Coordinate>](#) 라인 시작좌표, 끝좌표 Array

curve.shortestIntersectToLine(line) ⇒ [Array.<Coordinate>](#)

기하도형이 주어진 라인과 교차하는 좌표중 시작좌표에 가장 가까운 좌표를 반환한다.

Kind: instance method of [Curve](#)

Param Type Description

line [Array.<Coordinate>](#) 라인 시작좌표, 끝좌표 Array

curve.intersectLineToLine(line1, line2, extension) ⇒ [Coordinate](#)

라인1 과 라인2 의 교차점을 계산한다.

Kind: instance method of [Curve](#)

Returns: [Coordinate](#) - 교차점

Param	Type	Description
line1	<a href="#">Array.&lt;Coordinate&gt;</a>	line1 라인 시작좌표, 끝좌표 Array
line2	<a href="#">Array.&lt;Coordinate&gt;</a>	line2 라인 시작좌표, 끝좌표 Array
extension	boolean	라인을 연장하여 교차점을 계산하는 여부

curve.intersectCircleToLine(`center`, `radius`, `from`, `to`) ⇒ [Array.<Coordinate>](#)

주어진 원과 주어진 라인의 교차점을 계산한다.

Kind: instance method of [Curve](#)

Returns: [Array.<Coordinate>](#) - 교차점

Param	Type	Description
center	<a href="#">Coordinate</a>	중심점
radius	Number	반경
from	<a href="#">Coordinate</a>	라인 시작좌표
to	<a href="#">Coordinate</a>	라인 끝좌표

curve.intersectPointToLine(`p`, `line`) ⇒ [Coordinate](#)

포인트 P로부터 라인 AB 까지 수직인 가상선을 생각할때, 그 교차점을 계산한다.

Note: NON-ROBUST!

Kind: instance method of [Curve](#)

Returns: [Coordinate](#) - 교차점

Param	Type	Description
p	<a href="#">Coordinate</a>   <a href="#">Array.&lt;Number&gt;</a>	기준좌표
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 시작좌표, 끝좌표 Array

```
curve.getPercentageDistanceFromPoint(_coordinate) => Object
```

주어진 좌표에 대해 공간기하객체 바운더리 대비 가로, 세로 위치 퍼센테이지 비율을 구한다.

Kind: instance method of [Curve](#)

Returns: Object - {px , py}

Param	Type	Description
_coordinate	<a href="#">Coordinate</a>	좌표

```
curve.isContainsPoint(_coordinate) => boolean
```

공간기하객체가 주어진 좌표를 포함하는지를 반환한다.

Kind: instance method of [Curve](#)

Returns: boolean - true, false

Param	Type	Description
_coordinate	<a href="#">Coordinate</a>	좌표

```
curve.getPointFromPercentageDistance(pXpY) => Coordinate
```

공간기하객체에 대한 퍼센테이지 좌표의 실제 좌표를 구한다.

Kind: instance method of [Curve](#)

Returns: [Coordinate](#) - 실 좌표

Param	Type	Description
-------	------	-------------

## pXpY Array 퍼센테이지 좌표

curve.getParallelLine(from, to, distance) ⇒ [Array.<Coordinate>](#)

주어진 선분과 일정 거리에 있는 평행한 선분을 반환한다.

Kind: instance method of [Curve](#)

Returns: [Array.<Coordinate>](#) - 평행선 시작좌표, 끝좌표 Array

Param	Type	Description
from	<a href="#">Coordinate</a>	라인 시작좌표
to	<a href="#">Coordinate</a>	라인 끝좌표
distance		

curve.getParallelPath(line, distance)

주어진 라인과 일정 거리에 있는 평행한 라인을 반환한다.

Kind: instance method of [Curve](#)

Param	Type	Description
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 좌표 Array
distance		

curve.reset()

저장된 boundary 를 클리어하여 새로 계산하도록 한다.

Kind: instance method of [Curve](#)

geometry.Ellipse ⇐ [Curve](#)

Kind: static class of [geometry](#)

Extends: [Curve](#)

Requires: module:OG.geometry.Coordinate, module:OG.geometry.Envelope, module:OG.geometry.Geometry

Author: [Seungpil Park \(mailto:sppark@uengine.org\)](mailto:Seungpil.Park@sppark@uengine.org)

- [.Ellipse](#) ⇐ [Curve](#)

- [new OG.geometry.Ellipse\(center, radiusX, radiusY, angle\)](#)
- [.vertices : Array.<Coordinate>](#)
- [.TYPE : Number](#)
- [.IS\\_CLOSED : Boolean](#)
- [.style : Style](#)
- [.boundary : Envelope](#)
- [.getControlPoints\(\) ⇒ Array.<Coordinate>](#)
- [.getVertices\(\) ⇒ Array.<Coordinate>](#)
- [.toString\(\) ⇒ String](#)
- [.move\(offsetX, offsetY\) ⇒ Geometry](#)
- [.resize\(upper, lower, left, right\) ⇒ Geometry](#)
- [.rotate\(angle, origin\) ⇒ Geometry](#)
- [.angleBetweenPoints\(prev, next\) ⇒ Number](#)
- [.isRightAngleBetweenPoints\(prev, next\) ⇒ Object](#)
- [.angleBetweenThreePoints\(prev, next\) ⇒ Number](#)
- [.isEqual\(\\_geometry\) ⇒ Boolean](#)

- [.isContains\(\\_geometry\)](#) ⇒ Boolean
- [.isWithin\(\\_geometry\)](#) ⇒ Boolean
- [.getBoundary\(\)](#) ⇒ [Envelope](#)
- [.getCentroid\(\)](#) ⇒ [Coordinate](#)
- [.minDistance\(\\_coordinate\)](#) ⇒ Number
- [.distance\(\\_geometry\)](#) ⇒ Number
- [.getLength\(\)](#) ⇒ Number
- [.moveCentroid\(중심\)](#)
- [.resizeBox\(width, height\)](#) ⇒ [Geometry](#)
- [.fitToBoundary\(envelope\)](#) ⇒ [Geometry](#)
- [.convertCoordinate\(coordinate\)](#) ⇒ [Coordinate](#)
- [.distanceToLine\(p, line\)](#) ⇒ Number
- [.distanceLineToLine\(line1, line2\)](#) ⇒ Number
- [.intersectToLine\(line\)](#) ⇒ [Array.<Coordinate>](#)
- [.shortestIntersectToLine\(line\)](#) ⇒ [Array.<Coordinate>](#)
- [.intersectLineToLine\(line1, line2, extension\)](#) ⇒ [Coordinate](#)
- [.intersectCircleToLine\(center, radius, from, to\)](#) ⇒ [Array.<Coordinate>](#)
- [.intersectPointToLine\(p, line\)](#) ⇒ [Coordinate](#)
- [.getPercentageDistanceFromPoint\(\\_coordinate\)](#) ⇒ Object
- [.isContainsPoint\(\\_coordinate\)](#) ⇒ boolean
- [.getPointFromPercentageDistance\(pXpY\)](#) ⇒ [Coordinate](#)
- [.getParallelLine\(from, to, distance\)](#) ⇒ [Array.<Coordinate>](#)
- [.getParallelPath\(line, distance\)](#)
- [.reset\(\)](#)

```
new OG.geometry.Ellipse(center, radiusX, radiusY, angle)
```

`Ellipse` 공간 기하 객체(Spatial Geometry Object)

Param	Type	Description
center	<a href="#">Coordinate</a>	<code>Ellipse</code> 중심 좌표
radiusX	Number	X축 반경
radiusY	Number	Y축 반경
angle	Number	X축 기울기

Example

```
var geom = new OG.geometry.Ellipse([10, 10], 10, 5);
```

```
ellipse.vertices : Array.<Coordinate>
```

Line Vertex 좌표 Array

Kind: instance property of [Ellipse](#)

```
ellipse.TYPE : Number
```

공간 기하 객체 타입

Kind: instance property of [Ellipse](#)

Overrides: [TYPE](#)

```
ellipse.IS_CLOSED : Boolean
```

닫힌 기하 객체 인지 여부

Kind: instance property of [Ellipse](#)

Overrides: [IS\\_CLOSED](#)

```
ellipse.style : Style
```

스타일 속성

Kind: instance property of [Ellipse](#)

Overrides: [style](#)

```
ellipse.boundary : Envelope
```

공간기하객체를 포함하는 사각형의 Boundary 영역

Kind: instance property of [Ellipse](#)

```
ellipse.getControlPoints() ⇒ Array.<Coordinate>
```

콘트롤 포인트 목록을 반환한다.

Kind: instance method of [Ellipse](#)

Overrides: [getControlPoints](#)

Returns: [Array.<Coordinate>](#) - controlPoints Array

```
ellipse.getVertices() => Array.<Coordinate>
```

공간기하 객체의 모든 꼭지점을 반환한다.

Kind: instance method of [Ellipse](#)

Overrides: [getVertices](#)

Returns: [Array.<Coordinate>](#) - 꼭지점 좌표 Array

```
ellipse.toString() => String
```

객체 프라퍼티 정보를 JSON 스트링으로 반환한다.

Kind: instance method of [Ellipse](#)

Overrides: [toString](#)

Returns: String - 프라퍼티 정보

```
ellipse.move(offsetX, offsetY) => Geometry
```

가로, 세로 Offset 만큼 좌표를 이동한다.

Kind: instance method of [Ellipse](#)

Returns: [Geometry](#) - 이동된 공간 기하 객체

Param	Type	Description
offsetX	Number	가로 Offset
offsetY	Number	세로 Offset

```
ellipse.resize(upper, lower, left, right) ⇒ Geometry
```

상, 하, 좌, 우 외곽선을 이동하여 Envelope 을 리사이즈 한다.

Kind: instance method of [Ellipse](#)

Returns: [Geometry](#) - 리사이즈된 공간 기하 객체

Param	Type	Description
-------	------	-------------

upper Number 상단 라인 이동 Offset(위 방향으로 +)

lower Number 하단 라인 이동 Offset(아래 방향으로 +)

left Number 좌측 라인 이동 Offset(좌측 방향으로 +)

right Number 우측 라인 이동 Offset(우측 방향으로 +)

```
ellipse.rotate(angle, origin) ⇒ Geometry
```

기준 좌표를 기준으로 주어진 각도 만큼 회전한다.

Kind: instance method of [Ellipse](#)

Returns: [Geometry](#) - 회전된 공간 기하 객체

Param	Type	Description
-------	------	-------------

angle Number 회전 각도

origin [Coordinate](#) 기준 좌표(default:중심좌표)

```
ellipse.angleBetweenPoints(prev, next) ⇒ Number
```

공간기하객체의 두 꼭지점 사이에 가상의 선을 그렸을때, 그 기울기를 구한다.

Kind: instance method of [Ellipse](#)

Returns: Number - 기울기

Param Type Description

prev [Coordinate](#) 꼭지점 1

next [Coordinate](#) 꼭지점 2

```
ellipse.isRightAngleBetweenPoints(prev, next) => Object
```

공간기하객체의 두 꼭지점 사이의 기울기가 수평또는 수직인지 판별한다.

Kind: instance method of [Ellipse](#)

Returns: Object - {flag : true or false, type: horizontal or vertical or none}

Param Type Description

prev [Coordinate](#) 꼭지점 1

next [Coordinate](#) 꼭지점 2

```
ellipse.angleBetweenThreePoints(prev, next) => Number
```

공간기하객체의 세 꼭지점 사이의 각도 중 작은 각도를 반환한다.

Kind: instance method of [Ellipse](#)

Returns: Number - 기울기

Param Type Description

prev [Coordinate](#) 꼭지점 1

next [Coordinate](#) 꼭지점 2

```
ellipse.equals(_geometry) => Boolean
```

주어진 Geometry 객체와 같은지 비교한다.

Kind: instance method of [Ellipse](#)

Returns: Boolean - true:같음, false:다름

Param	Type	Description
_geometry	<a href="#">Geometry</a>	Geometry 객체

```
ellipse.isContains(_geometry) ⇒ Boolean
```

주어진 공간기하객체를 포함하는지 비교한다.

Kind: instance method of [Ellipse](#)

Returns: Boolean - 포함하면 true

Param	Type	Description
_geometry	<a href="#">Geometry</a>	Geometry 객체

```
ellipse.isWithin(_geometry) ⇒ Boolean
```

주어진 공간기하객체에 포함되는지 비교한다.

Kind: instance method of [Ellipse](#)

Returns: Boolean - 포함되면 true

Param	Type	Description
_geometry	<a href="#">Geometry</a>	Geometry 객체

```
ellipse.getBoundary() ⇒ Envelope
```

공간기하객체를 포함하는 사각형의 Boundary 영역을 반환한다.

Kind: instance method of [Ellipse](#)

Returns: [Envelope](#) - Envelope 영역

`ellipse.getCentroid() ⇒ Coordinate`

공간기하객체의 중심좌표를 반환한다.

Kind: instance method of [Ellipse](#)

Returns: [Coordinate](#) - 중심좌표

`ellipse.minDistance(_coordinate) ⇒ Number`

주어진 좌표와의 최단거리를 반환한다.

Kind: instance method of [Ellipse](#)

Returns: Number - 최단거리

Param	Type	Description
<code>_coordinate</code>	<a href="#">Coordinate</a>	좌표

`ellipse.distance(_geometry) ⇒ Number`

주어진 공간기하객체와의 중심점 간의 거리를 반환한다.

Kind: instance method of [Ellipse](#)

Returns: Number - 거리

Param	Type	Description
_geometry	<a href="#">Geometry</a>	공간 기하 객체

ellipse.getLength() ⇒ Number

공간기하객체의 길이를 반환한다.

Kind: instance method of [Ellipse](#)

Overrides: [getLength](#)

Returns: Number - 길이

ellipse.moveCentroid(중심)

주어진 중심좌표로 공간기하객체를 이동한다.

Kind: instance method of [Ellipse](#)

Param	Type	Description
중심	<a href="#">Coordinate</a>	좌표

ellipse.resizeBox(width, height) ⇒ [Geometry](#)

중심좌표는 고정한 채 Bounding Box 의 width, height 를 리사이즈 한다.

Kind: instance method of [Ellipse](#)

Returns: [Geometry](#) - 리사이즈된 공간 기하 객체

Param	Type	Description
width	Number	너비

height Number 높이

ellipse.fitToBoundary(envelope) ⇒ [Geometry](#)

주어진 Boundary 영역 안으로 공간 기하 객체를 적용한다.(이동 & 리사이즈)

Kind: instance method of [Ellipse](#)

Returns: [Geometry](#) - 적용된 공간 기하 객체

Param	Type	Description
envelope	<a href="#">Envelope</a>	Envelope 영역

ellipse.convertCoordinate(coordinate) ⇒ [Coordinate](#)

파라미터가 [x, y] 형식의 좌표 Array 이면 OG.geometry.Coordinate 인스턴스를 new 하여 반환한다.

Kind: instance method of [Ellipse](#)

Param	Type	Description
coordinate	<a href="#">Coordinate</a>   Array. <a href="#">Number</a>	[x, y] 형식의 좌표 Array 또는 OG.geometry.Coordinate 인스턴스

ellipse.distanceToLine(p, line) ⇒ Number

포인트 P로부터 라인 AB의 거리를 계산한다.

Note: NON-ROBUST!

Kind: instance method of [Ellipse](#)

Returns: Number - 거리

Param	Type	Description
p	<a href="#">Coordinate</a>   Array.<Number>	기준좌표
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 시작좌표, 끝좌표 Array

ellipse.distanceLineToLine(line1, line2) ⇒ Number

라인1로부터 라인2의 거리를 계산한다.

Note: NON-ROBUST!

Kind: instance method of [Ellipse](#)

Returns: Number - 거리

Param	Type	Description
line1	<a href="#">Array.&lt;Coordinate&gt;</a>	line1 라인 시작좌표, 끝좌표 Array
line2	<a href="#">Array.&lt;Coordinate&gt;</a>	line2 라인 시작좌표, 끝좌표 Array

ellipse.intersectToLine(line) ⇒ [Array.<Coordinate>](#)

기하도형이 주어진 라인과 교차하는 좌표들을 반환한다.

Kind: instance method of [Ellipse](#)

Param	Type	Description
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 시작좌표, 끝좌표 Array

ellipse.shortestIntersectToLine(line) ⇒ [Array.<Coordinate>](#)

기하도형이 주어진 라인과 교차하는 좌표중 시작좌표에 가장 가까운 좌표를 반환한다.

Kind: instance method of [Ellipse](#)

Param	Type	Description
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 시작좌표, 끝좌표 Array

```
ellipse.intersectLineToLine(line1, line2, extension) ⇒ Coordinate
```

라인1 과 라인2 의 교차점을 계산한다.

Kind: instance method of [Ellipse](#)

Returns: [Coordinate](#) - 교차점

Param	Type	Description
line1	<a href="#">Array.&lt;Coordinate&gt;</a>	line1 라인 시작좌표, 끝좌표 Array
line2	<a href="#">Array.&lt;Coordinate&gt;</a>	line2 라인 시작좌표, 끝좌표 Array
extension	boolean	라인을 연장하여 교차점을 계산하는 여부

```
ellipse.intersectCircleToLine(center, radius, from, to) ⇒ Array.<Coordinate>
```

주어진 원과 주어진 라인의 교차점을 계산한다.

Kind: instance method of [Ellipse](#)

Returns: [Array.<Coordinate>](#) - 교차점

Param	Type	Description
center	<a href="#">Coordinate</a>	중심점
radius	Number	반경
from	<a href="#">Coordinate</a>	line 라인 시작좌표
to	<a href="#">Coordinate</a>	line 라인 끝좌표

```
ellipse.intersectPointToLine(p, line) ⇒ Coordinate
```

포인트 P로부터 라인 AB 까지 수직인 가상선을 생각할때, 그 교차점을 계산한다.

Note: NON-ROBUST!

Kind: instance method of [Ellipse](#)

Returns: [Coordinate](#) - 교차점

Param	Type	Description
p	<a href="#">Coordinate</a>   Array.<Number>	기준좌표
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 시작좌표, 끝좌표 Array

```
ellipse.getPercentageDistanceFromPoint(_coordinate) => Object
```

주어진 좌표에 대해 공간기하객체 바운더리 대비 가로, 세로 위치 퍼센테이지 비율을 구한다.

Kind: instance method of [Ellipse](#)

Returns: Object - {px , py}

Param	Type	Description
_coordinate	<a href="#">Coordinate</a>	좌표

```
ellipse.isContainsPoint(_coordinate) => boolean
```

공간기하객체가 주어진 좌표를 포함하는지를 반환한다.

Kind: instance method of [Ellipse](#)

Returns: boolean - true, false

Param	Type	Description
_coordinate	<a href="#">Coordinate</a>	좌표

```
ellipse.getPointFromPercentageDistance(pXpY) ⇒ Coordinate
```

공간기하객체에 대한 퍼센테이지 좌표의 실제 좌표를 구한다.

Kind: instance method of [Ellipse](#)

Returns: [Coordinate](#) - 실 좌표

Param Type Description

pXpY Array 퍼센테이지 좌표

```
ellipse.getParallelLine(from, to, distance) ⇒ Array.<Coordinate>
```

주어진 선분과 일정 거리에 있는 평행한 선분을 반환한다.

Kind: instance method of [Ellipse](#)

Returns: [Array.<Coordinate>](#) - 평행선 시작좌표, 끝좌표 Array

Param	Type	Description
from	<a href="#">Coordinate</a>	라인 시작좌표
to	<a href="#">Coordinate</a>	라인 끝좌표
distance		

```
ellipse.getParallelPath(line, distance)
```

주어진 라인과 일정 거리에 있는 평행한 라인을 반환한다.

Kind: instance method of [Ellipse](#)

Param	Type	Description
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 좌표 Array

distance

```
ellipse.reset()
```

저장된 boundary 를 클리어하여 새로 계산하도록 한다.

Kind: instance method of [Ellipse](#)

geometry.Envelope

Kind: static class of [geometry](#)

Requires: module:OG.geometry.Coordinate

Author: [Seungpil Park \(mailto:sppark@uengine.org\)](mailto:Seungpil.Park@sppark@uengine.org)

- [.Envelope](#)

- [new OG.geometry.Envelope\(upperLeft, width, height\)](#)
- [.getUpperLeft\(\) ⇒ Coordinate](#)
- [.setUpperLeft\(upperLeft\)](#)
- [.getUpperRight\(\) ⇒ Coordinate](#)
- [.getLowerRight\(\) ⇒ Coordinate](#)
- [.getLowerLeft\(\) ⇒ Coordinate](#)
- [.getLeftCenter\(\) ⇒ Coordinate](#)
- [.getUpperCenter\(\) ⇒ Coordinate](#)
- [.getRightCenter\(\) ⇒ Coordinate](#)
- [.getLowerCenter\(\) ⇒ Coordinate](#)

- [.getCentroid\(\)](#) ⇒ [Coordinate](#)
- [.setCentroid\(centroid\)](#)
- [.getWidth\(\)](#) ⇒ Number
- [.setWidth\(width\)](#)
- [.getHeight\(\)](#) ⇒ Number
- [.setHeight\(height\)](#)
- [.getVertices\(\)](#) ⇒ [Array.<Coordinate>](#)
- [.isContains\(coordinate\)](#) ⇒ Boolean
- [.isContainsAll\(coordinateArray\)](#) ⇒ Boolean
- [.getHowManyContains\(coordinateArray\)](#) ⇒ Boolean
- [.isContainsOnce\(coordinateArray\)](#) ⇒ Boolean
- [.move\(offsetX, offsetY\)](#) ⇒ [Envelope](#)
- [.resize\(upper, lower, left, right\)](#) ⇒ [Envelope](#)
- [.equals\(Envelope\)](#) ⇒ Boolean
- [.toString\(\)](#) ⇒ String

```
new OG.geometry.Envelope(upperLeft, width, height)
```

2차원 좌표계에서 Envelope 영역을 정의

Param	Type	Description
upperLeft	<a href="#">Coordinate</a>   <a href="#">Array.&lt;Number&gt;</a>	기준 좌상단 좌표
width	Number	너비
height	Number	높이

Example

```
var boundingBox = new OG.geometry.Envelope([50, 50], 200, 100);
```

envelope.getUpperLeft() ⇒ [Coordinate](#)

기준 좌상단 좌표를 반환한다.

Kind: instance method of [Envelope](#)

Returns: [Coordinate](#) - 좌상단 좌표

envelope.setUpperLeft(upperLeft)

주어진 좌표로 기준 좌상단 좌표를 설정한다. 새로 설정된 값으로 이동된다.

Kind: instance method of [Envelope](#)

Param	Type	Description
upperLeft <a href="#">Coordinate</a>   Array.<Number>	좌상단 좌표	

envelope.getUpperRight() ⇒ [Coordinate](#)

우상단 좌표를 반환한다.

Kind: instance method of [Envelope](#)

Returns: [Coordinate](#) - 우상단 좌표

envelope.getLowerRight() ⇒ [Coordinate](#)

우하단 좌표를 반환한다.

Kind: instance method of [Envelope](#)

Returns: [Coordinate](#) - 우하단 좌표

envelope.getLowerLeft() ⇒ [Coordinate](#)

좌하단 좌표를 반환한다.

Kind: instance method of [Envelope](#)

Returns: [Coordinate](#) - 좌하단 좌표

envelope.getLeftCenter() ⇒ [Coordinate](#)

좌중간 좌표를 반환한다.

Kind: instance method of [Envelope](#)

Returns: [Coordinate](#) - 좌중간 좌표

envelope.getUpperCenter() ⇒ [Coordinate](#)

상단중간 좌표를 반환한다.

Kind: instance method of [Envelope](#)

Returns: [Coordinate](#) - 상단중간 좌표

envelope.getRightCenter() ⇒ [Coordinate](#)

우중간 좌표를 반환한다.

Kind: instance method of [Envelope](#)

Returns: [Coordinate](#) - 우중간 좌표

envelope.getLowerCenter() ⇒ [Coordinate](#)

하단중간 좌표를 반환한다.

Kind: instance method of [Envelope](#)

Returns: [Coordinate](#) - 하단중간 좌표

envelope.getCentroid() ⇒ [Coordinate](#)

Envelope 의 중심좌표를 반환한다.

Kind: instance method of [Envelope](#)

Returns: [Coordinate](#) - 중심좌표

envelope.setCentroid(centroid)

주어진 좌표로 중심 좌표를 설정한다. 새로 설정된 값으로 이동된다.

Kind: instance method of [Envelope](#)

Param	Type	Description
centroid	<a href="#">Coordinate</a>   Array.<Number>	중심좌표

```
envelope.getWidth() => Number
```

Envelope 의 가로 사이즈를 반환한다.

Kind: instance method of [Envelope](#)

Returns: Number - 너비

```
envelope.setWidth(width)
```

주어진 값으로 Envelope 의 가로 사이즈를 설정한다.

Kind: instance method of [Envelope](#)

Param Type Description

width Number 너비

```
envelope.getHeight() => Number
```

Envelope 의 세로 사이즈를 반환한다.

Kind: instance method of [Envelope](#)

Returns: Number - 높이

```
envelope.setHeight(height)
```

주어진 값으로 Envelope 의 세로 사이즈를 설정한다.

Kind: instance method of [Envelope](#)

Param Type Description

height Number 높이

`envelope.getVertices() ⇒ Array.<Coordinate>`

`Envelope` 모든 꼭지점을 반환한다.

좌상단좌표부터 시계방향으로 꼭지점 Array 를 반환한다.

Kind: instance method of [Envelope](#)

Returns: [Array.<Coordinate>](#) - 꼭지점 좌표 Array : [좌상단, 상단중간, 우상단, 우증간, 우하단, 하단증간, 좌하단, 좌증간, 좌상단]

`envelope.isContains(coordinate) ⇒ Boolean`

주어진 좌표값이 `Envelope` 영역에 포함되는지 비교한다.

Kind: instance method of [Envelope](#)

Returns: Boolean - true:포함, false:비포함

Param	Type	Description
coordinate	<a href="#">Coordinate</a>   <code>Array.&lt;Number&gt;</code>	좌표값

`envelope.isContainsAll(coordinateArray) ⇒ Boolean`

주어진 모든 좌표값이 `Envelope` 영역에 포함되는지 비교한다.

Kind: instance method of [Envelope](#)

Returns: Boolean - true:포함, false:비포함

Param	Type	Description
coordinateArray	<a href="#">Array.&lt;Coordinate&gt;</a>	좌표값 Array

envelope.getHowManyContains(coordinateArray) ⇒ Boolean

주어진 모든 좌표값이 Envelope 영역에 포함되는지 비교한다.

Kind: instance method of [Envelope](#)

Returns: Boolean - true:포함, false:비포함

Param	Type	Description
coordinateArray	<a href="#">Array.&lt;Coordinate&gt;</a>	좌표값 Array

envelope.isContainsOnce(coordinateArray) ⇒ Boolean

주어진 모든 좌표값이 Envelope 영역에 포함되는지 비교한다.

Kind: instance method of [Envelope](#)

Returns: Boolean - true:포함, false:비포함

Param	Type	Description
coordinateArray	<a href="#">Array.&lt;Coordinate&gt;</a>	좌표값 Array

envelope.move(offsetX, offsetY) ⇒ [Envelope](#)

크기는 고정한 채 가로, 세로 Offset 만큼 Envelope 을 이동한다.

Kind: instance method of [Envelope](#)

Returns: [Envelope](#) - 이동된 Envelope

Param	Type	Description
offsetX	Number	가로 Offset
offsetY	Number	세로 Offset

envelope.resize(upper, lower, left, right) ⇒ [Envelope](#)

상, 하, 좌, 우 외곽선을 이동하여 Envelope 을 리사이즈 한다.

Kind: instance method of [Envelope](#)

Returns: [Envelope](#) - 리사이즈된 Envelope

Param	Type	Description
upper	Number	상단 라인 이동 Offset(위 방향으로 +)
lower	Number	하단 라인 이동 Offset(아래 방향으로 +)
left	Number	좌측 라인 이동 Offset(좌측 방향으로 +)
right	Number	우측 라인 이동 Offset(우측 방향으로 +)

envelope.equals(Envelope) ⇒ Boolean

주어진 Envelope 영역과 같은지 비교한다.

Kind: instance method of [Envelope](#)

Returns: Boolean - true:같음, false:다름

Param	Type	Description
Envelope	<a href="#">Envelope</a>	영역

```
envelope.toString() => String
```

객체 프라퍼티 정보를 JSON 스트링으로 반환한다.

Kind: instance method of [Envelope](#)

Returns: String - 프라퍼티 정보

geometry.Geometry

Kind: static class of [geometry](#)

Requires: module:OG.geometry.Coordinate, module:OG.geometry.Envelope

Author: [Seungpil Park \(mailto:spspark@uengine.org\)](mailto:Seungpil.Park@uengine.org)

- [.Geometry](#)

- [new OG.geometry.Geometry\(\)](#)
- [.TYPE](#) : Number
- [.IS\\_CLOSED](#) : Boolean
- [.style](#) : [Style](#)
- [.boundary](#) : [Envelope](#)
- [.isEquals\(\\_geometry\)](#) => Boolean
- [.isContains\(\\_geometry\)](#) => Boolean
- [.isWithin\(\\_geometry\)](#) => Boolean
- [.getBoundary\(\)](#) => [Envelope](#)
- [.getCentroid\(\)](#) => [Coordinate](#)
- [.getVertices\(\)](#) => [Array.<Coordinate>](#)
- [.minDistance\(\\_coordinate\)](#) => Number

- [.distance\(\\_geometry\)](#) ⇒ Number
- [.getLength\(\)](#) ⇒ Number
- [.move\(offsetX, offsetY\)](#) ⇒ [Geometry](#)
- [.moveCentroid\(중심\)](#)
- [.resize\(upper, lower, left, right\)](#) ⇒ [Geometry](#)
- [.resizeBox\(width, height\)](#) ⇒ [Geometry](#)
- [.rotate\(angle, origin\)](#) ⇒ [Geometry](#)
- [.fitToBoundary\(envelope\)](#) ⇒ [Geometry](#)
- [.convertCoordinate\(coordinate\)](#) ⇒ [Coordinate](#)
- [.distanceToLine\(p, line\)](#) ⇒ Number
- [.distanceLineToLine\(line1, line2\)](#) ⇒ Number
- [.intersectToLine\(line\)](#) ⇒ [Array.<Coordinate>](#)
- [.shortestIntersectToLine\(line\)](#) ⇒ [Array.<Coordinate>](#)
- [.intersectLineToLine\(line1, line2, extension\)](#) ⇒ [Coordinate](#)
- [.intersectCircleToLine\(center, radius, from, to\)](#) ⇒ [Array.<Coordinate>](#)
- [.intersectPointToLine\(p, line\)](#) ⇒ [Coordinate](#)
- [.getPercentageDistanceFromPoint\(\\_coordinate\)](#) ⇒ Object
- [.isContainsPoint\(\\_coordinate\)](#) ⇒ boolean
- [.getPointFromPercentageDistance\(pXpY\)](#) ⇒ [Coordinate](#)
- [.getParallelLine\(from, to, distance\)](#) ⇒ [Array.<Coordinate>](#)
- [.getParallelPath\(line, distance\)](#)
- [.reset\(\)](#)

```
new OG.geometry.Geometry()
```

공간 기하 객체(Spatial Geometry Object)의 최상위 추상 클래스

```
geometry.TYPE : Number
```

공간 기하 객체 타입

Kind: instance property of [Geometry](#)

```
geometry.IS_CLOSED : Boolean
```

닫힌 기하 객체 인지 여부

Kind: instance property of [Geometry](#)

```
geometry.style : Style
```

스타일 속성

Kind: instance property of [Geometry](#)

```
geometry.boundary : Envelope
```

공간기하객체를 포함하는 사각형의 Boundary 영역

Kind: instance property of [Geometry](#)

`geometry.equals(_geometry) ⇒ Boolean`

주어진 Geometry 객체와 같은지 비교한다.

Kind: instance method of [Geometry](#)

Returns: Boolean - true:같음, false:다름

Param	Type	Description
<code>_geometry</code>	<a href="#">Geometry</a>	Geometry 객체

`geometry.contains(_geometry) ⇒ Boolean`

주어진 공간기하객체를 포함하는지 비교한다.

Kind: instance method of [Geometry](#)

Returns: Boolean - 포함하면 true

Param	Type	Description
<code>_geometry</code>	<a href="#">Geometry</a>	Geometry 객체

`geometry.within(_geometry) ⇒ Boolean`

주어진 공간기하객체에 포함되는지 비교한다.

Kind: instance method of [Geometry](#)

Returns: Boolean - 포함되면 true

Param	Type	Description
_geometry	<a href="#">Geometry</a>	Geometry 객체

`geometry.getBoundary() ⇒ Envelope`

공간기하객체를 포함하는 사각형의 Boundary 영역을 반환한다.

Kind: instance method of [Geometry](#)

Returns: [Envelope](#) - Envelope 영역

`geometry.getCentroid() ⇒ Coordinate`

공간기하객체의 중심좌표를 반환한다.

Kind: instance method of [Geometry](#)

Returns: [Coordinate](#) - 중심좌표

`geometry.getVertices() ⇒ Array.<Coordinate>`

공간기하객체의 모든 꼭지점을 반환한다.

Kind: instance abstract method of [Geometry](#)

Returns: [Array.<Coordinate>](#) - 꼭지점 좌표 Array

`geometry.minDistance(_coordinate) ⇒ Number`

주어진 좌표와의 최단거리를 반환한다.

Kind: instance method of [Geometry](#)

Returns: Number - 최단거리

Param	Type	Description
_coordinate	<a href="#">Coordinate</a>	좌표

```
geometry.distance(_geometry) => Number
```

주어진 공간기하 객체와의 중심점 간의 거리를 반환한다.

Kind: instance method of [Geometry](#)

Returns: Number - 거리

Param	Type	Description
geometry	<a href="#">Geometry</a>	공간 기하 객체

```
geometry.getLength() => Number
```

공간기하 객체의 길이를 반환한다.

Kind: instance method of [Geometry](#)

Returns: Number - 길이

```
geometry.move(offsetX, offsetY) => Geometry
```

가로, 세로 Offset 만큼 좌표를 이동한다.

Kind: instance abstract method of [Geometry](#)

Returns: [Geometry](#) - 이동된 공간 기하 객체

Param Type Description  
offsetX Number 가로 Offset  
offsetY Number 세로 Offset

`geometry.moveCentroid(중심)`

주어진 중심좌표로 공간기하객체를 이동한다.

Kind: instance method of [Geometry](#)

Param Type Description  
중심 [Coordinate](#) 좌표

`geometry.resize(upper, lower, left, right) ⇒ Geometry`

상, 하, 좌, 우 외곽선을 이동하여 Envelope 을 리사이즈 한다.

Kind: instance abstract method of [Geometry](#)

Returns: [Geometry](#) - 리사이즈된 공간 기하 객체

Param Type Description  
upper Number 상단 라인 이동 Offset(위 방향으로 +)  
lower Number 하단 라인 이동 Offset(아래 방향으로 +)  
left Number 좌측 라인 이동 Offset(좌측 방향으로 +)  
right Number 우측 라인 이동 Offset(우측 방향으로 +)

`geometry.resizeBox(width, height) ⇒ Geometry`

중심좌표는 고정한 채 Bounding Box 의 width, height 를 리사이즈 한다.

Kind: instance method of [Geometry](#)

Returns: [Geometry](#) - 리사이즈된 공간 기하 객체

Param Type Description

width Number 너비

height Number 높이

`geometry.rotate(angle, origin) ⇒ Geometry`

기준 좌표를 기준으로 주어진 각도 만큼 회전한다.

Kind: instance abstract method of [Geometry](#)

Returns: [Geometry](#) - 회전된 공간 기하 객체

Param Type Description

angle Number 회전 각도

origin [Coordinate](#) 기준 좌표(default:중심좌표)

`geometry.fitToBoundary(envelope) ⇒ Geometry`

주어진 Boundary 영역 안으로 공간 기하 객체를 적용한다.(이동 & 리사이즈)

Kind: instance method of [Geometry](#)

Returns: [Geometry](#) - 적용된 공간 기하 객체

Param Type Description

envelope [Envelope](#) Envelope 영역

```
geometry.convertCoordinate(coordinate) ⇒ Coordinate
```

파라미터가 [x, y] 형식의 좌표 Array 이면 OG.geometry.Coordinate 인스턴스를 new 하여 반환한다.

Kind: instance method of [Geometry](#)

Param	Type	Description
coordinate	<a href="#">Coordinate</a>   Array.<Number>	[x, y] 형식의 좌표 Array 또는 OG.geometry.Coordinate 인스턴스

```
geometry.distanceToLine(p, line) ⇒ Number
```

포인트 P로부터 라인 AB의 거리를 계산한다.

Note: NON-ROBUST!

Kind: instance method of [Geometry](#)

Returns: Number - 거리

Param	Type	Description
p	<a href="#">Coordinate</a>   Array.<Number>	기준좌표
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 시작좌표, 끝좌표 Array

```
geometry.distanceLineToLine(line1, line2) ⇒ Number
```

라인1로부터 라인2의 거리를 계산한다.

Note: NON-ROBUST!

Kind: instance method of [Geometry](#)

Returns: Number - 거리

Param	Type	Description
line1	<a href="#">Array.&lt;Coordinate&gt;</a>	line1 라인 시작좌표, 끝좌표 Array
line2	<a href="#">Array.&lt;Coordinate&gt;</a>	line2 라인 시작좌표, 끝좌표 Array

`geometry.intersectToLine(line) ⇒ Array.<Coordinate>`

기하도형이 주어진 라인과 교차하는 좌표들을 반환한다.

Kind: instance method of [Geometry](#)

Param	Type	Description
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 시작좌표, 끝좌표 Array

`geometry.shortestIntersectToLine(line) ⇒ Array.<Coordinate>`

기하도형이 주어진 라인과 교차하는 좌표중 시작좌표에 가장 가까운 좌표를 반환한다.

Kind: instance method of [Geometry](#)

Param	Type	Description
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 시작좌표, 끝좌표 Array

`geometry.intersectLineToLine(line1, line2, extension) ⇒ Coordinate`

라인1 과 라인2 의 교차점을 계산한다.

Kind: instance method of [Geometry](#)

Returns: [Coordinate](#) - 교차점

Param	Type	Description
line1	<a href="#">Array.&lt;Coordinate&gt;</a>	line1 라인 시작좌표, 끝좌표 Array
line2	<a href="#">Array.&lt;Coordinate&gt;</a>	line2 라인 시작좌표, 끝좌표 Array
extension	boolean	라인을 연장하여 교차점을 계산하는 여부

`geometry.intersectCircleToLine(center, radius, from, to) ⇒ Array.<Coordinate>`

주어진 원과 주어진 라인의 교차점을 계산한다.

Kind: instance method of [Geometry](#)

Returns: [Array.<Coordinate>](#) - 교차점

Param	Type	Description
center	<a href="#">Coordinate</a>	중심점
radius	Number	반경
from	<a href="#">Coordinate</a>	line 라인 시작좌표
to	<a href="#">Coordinate</a>	line 라인 끝좌표

`geometry.intersectPointToLine(p, line) ⇒ Coordinate`

포인트 P로부터 라인 AB 까지 수직인 가상선을 생각할때, 그 교차점을 계산한다.

Note: NON-ROBUST!

Kind: instance method of [Geometry](#)

Returns: [Coordinate](#) - 교차점

Param	Type	Description
p	<a href="#">Coordinate</a>   <a href="#">Array.&lt;Number&gt;</a>	기준좌표
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 시작좌표, 끝좌표 Array

`geometry.getPercentageDistanceFromPoint(_coordinate) ⇒ Object`

주어진 좌표에 대해 공간기하객체 바운더리 대비 가로, 세로 위치 퍼센테이지 비율을 구한다.

Kind: instance method of [Geometry](#)

Returns: Object - {px , py}

Param	Type	Description
_coordinate	<a href="#">Coordinate</a>	좌표

```
geometry.isContainsPoint(_coordinate) => boolean
```

공간기하객체가 주어진 좌표를 포함하는지를 반환한다.

Kind: instance method of [Geometry](#)

Returns: boolean - true, false

Param	Type	Description
_coordinate	<a href="#">Coordinate</a>	좌표

```
geometry.getPointFromPercentageDistance(pXpY) => Coordinate
```

공간기하객체에 대한 퍼센테이지 좌표의 실제 좌표를 구한다.

Kind: instance method of [Geometry](#)

Returns: [Coordinate](#) - 실 좌표

Param	Type	Description
pXpY	Array	퍼센테이지 좌표

```
geometry.getParallelLine(from, to, distance) => Array.<Coordinate>
```

주어진 선분과 일정 거리에 있는 평행한 선분을 반환한다.

Kind: instance method of [Geometry](#)

Returns: [Array.<Coordinate>](#) - 평행선 시작좌표, 끝좌표 Array

Param	Type	Description
from	<a href="#">Coordinate</a>	라인 시작좌표
to	<a href="#">Coordinate</a>	라인 끝좌표
distance		

```
geometry.getParallelPath(line, distance)
```

주어진 라인과 일정 거리에 있는 평행한 라인을 반환한다.

Kind: instance method of [Geometry](#)

Param	Type	Description
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 좌표 Array
distance		

```
geometry.reset()
```

저장된 boundary 를 클리어하여 새로 계산하도록 한다.

Kind: instance method of [Geometry](#)

```
geometry.GeometryCollection ← Geometry
```

Kind: static class of [geometry](#)

Extends: [Geometry](#)

Requires: module:OG.geometry.Coordinate, module:OG.geometry.Envelope, module:OG.geometry.Geometry

Author: [Seungpil Park \(mailto:spspark@uengine.org\)](mailto:Seungpil.Park@uengine.org)

- [.GeometryCollection](#) ⇐ [Geometry](#)
  - [new OG.geometry.GeometryCollection\(geometries\)](#)
  - [.geometries](#) : [Array.<Geometry>](#)
  - [.TYPE](#) : Number
  - [.IS\\_CLOSED](#) : Boolean
  - [.style](#) : [Style](#)
  - [.boundary](#) : [Envelope](#)
  - [.toString\(\)](#) ⇒ String
  - [.isEquals\( geometry \)](#) ⇒ Boolean
  - [.isContains\( geometry \)](#) ⇒ Boolean
  - [.isWithin\( geometry \)](#) ⇒ Boolean
  - [.getBoundary\(\)](#) ⇒ [Envelope](#)
  - [.getCentroid\(\)](#) ⇒ [Coordinate](#)
  - [.getVertices\(\)](#) ⇒ [Array.<Coordinate>](#)
  - [.minDistance\( coordinate \)](#) ⇒ Number
  - [.distance\( geometry \)](#) ⇒ Number
  - [.getLength\(\)](#) ⇒ Number
  - [.move\(offsetX, offsetY\)](#) ⇒ [Geometry](#)
  - [.moveCentroid\(중심\)](#)
  - [.resize\(upper, lower, left, right\)](#) ⇒ [Geometry](#)

- [.resizeBox\(width, height\)](#) ⇒ [Geometry](#)
- [.rotate\(angle, origin\)](#) ⇒ [Geometry](#)
- [.fitToBoundary\(envelope\)](#) ⇒ [Geometry](#)
- [.convertCoordinate\(coordinate\)](#) ⇒ [Coordinate](#)
- [.distanceToLine\(p, line\)](#) ⇒ Number
- [.distanceLineToLine\(line1, line2\)](#) ⇒ Number
- [.intersectToLine\(line\)](#) ⇒ [Array.<Coordinate>](#)
- [.shortestIntersectToLine\(line\)](#) ⇒ [Array.<Coordinate>](#)
- [.intersectLineToLine\(line1, line2, extension\)](#) ⇒ [Coordinate](#)
- [.intersectCircleToLine\(center, radius, from, to\)](#) ⇒ [Array.<Coordinate>](#)
- [.intersectPointToLine\(p, line\)](#) ⇒ [Coordinate](#)
- [.getPercentageDistanceFromPoint\(\\_coordinate\)](#) ⇒ Object
- [.isContainsPoint\(\\_coordinate\)](#) ⇒ boolean
- [.getPointFromPercentageDistance\(pXpY\)](#) ⇒ [Coordinate](#)
- [.getParallelLine\(from, to, distance\)](#) ⇒ [Array.<Coordinate>](#)
- [.getParallelPath\(line, distance\)](#)
- [.reset\(\)](#)

new OG.geometry.GeometryCollection(geometries)

공간 기하 객체(Spatial Geometry Object) Collection

Param	Type	Description
geometries	<a href="#"><u>Array.&lt;Geometry&gt;</u></a>	공간 기하 객체 Array

## Example

```
var geom1 = new OG.geometry.Point([20, 5]),  
    geom2 = new OG.geometry.Line([20, 5], [30, 15]),  
    geom3 = new OG.geometry.PolyLine([[20, 5], [30, 15], [40, 25], [50, 15]]);  
  
var collection = new OG.geometry.GeometryCollection([geom1, geom2, geom3]);
```

geometryCollection.geometries : [Array.<Geometry>](#)

공간 기하 객체 Array

Kind: instance property of [GeometryCollection](#)

geometryCollection.TYPE : Number

공간 기하 객체 타입

Kind: instance property of [GeometryCollection](#)

Overrides: [TYPE](#)

geometryCollection.IS\_CLOSED : Boolean

닫힌 기하 객체 인지 여부

Kind: instance property of [GeometryCollection](#)

geometryCollection.style : [Style](#)

스타일 속성

Kind: instance property of [GeometryCollection](#)

Overrides: [style](#)

geometryCollection.boundary : [Envelope](#)

공간기하 객체를 포함하는 사각형의 Boundary 영역

Kind: instance property of [GeometryCollection](#)

geometryCollection.toString() ⇒ String

객체 프라퍼티 정보를 JSON 스트링으로 반환한다.

Kind: instance method of [GeometryCollection](#)

Returns: String - 프라퍼티 정보

geometryCollection.equals(\_geometry) ⇒ Boolean

주어진 Geometry 객체와 같은지 비교한다.

Kind: instance method of [GeometryCollection](#)

Returns: Boolean - true:같음, false:다름

Param	Type	Description
_geometry	<a href="#">Geometry</a>	Geometry 객체

```
geometryCollection.isContains(_geometry) ⇒ Boolean
```

주어진 공간기하 객체를 포함하는지 비교한다.

Kind: instance method of [GeometryCollection](#)

Returns: Boolean - 포함하면 true

Param	Type	Description
_geometry	<a href="#">Geometry</a>	Geometry 객체

```
geometryCollection.isWithin(_geometry) ⇒ Boolean
```

주어진 공간기하 객체에 포함되는지 비교한다.

Kind: instance method of [GeometryCollection](#)

Returns: Boolean - 포함되면 true

Param	Type	Description
_geometry	<a href="#">Geometry</a>	Geometry 객체

```
geometryCollection.getBoundary() ⇒ Envelope
```

공간기하 객체를 포함하는 사각형의 Boundary 영역을 반환한다.

Kind: instance method of [GeometryCollection](#)

Returns: [Envelope](#) - Envelope 영역

```
geometryCollection.getCentroid() => Coordinate
```

공간기하객체의 중심좌표를 반환한다.

Kind: instance method of [GeometryCollection](#)

Returns: [Coordinate](#) - 중심좌표

```
geometryCollection.getVertices() => Array.<Coordinate>
```

공간기하객체의 모든 꼭지점을 반환한다.

Kind: instance method of [GeometryCollection](#)

Overrides: [getVertices](#)

Returns: [Array.<Coordinate>](#) - 꼭지점 좌표 Array

```
geometryCollection.minDistance(_coordinate) => Number
```

주어진 좌표와의 최단거리를 반환한다.

Kind: instance method of [GeometryCollection](#)

Returns: Number - 최단거리

Param	Type	Description
_coordinate	<a href="#">Coordinate</a>	좌표

```
geometryCollection.distance(_geometry) => Number
```

주어진 공간기하객체와의 중심점 간의 거리를 반환한다.

Kind: instance method of [GeometryCollection](#)

Returns: Number - 거리

Param Type Description  
geometry [Geometry](#) 공간 기하 객체

geometryCollection.getLength() ⇒ Number

공간기하객체의 길이를 반환한다.

Kind: instance method of [GeometryCollection](#)

Returns: Number - 길이

geometryCollection.move(offsetX, offsetY) ⇒ [Geometry](#)

가로, 세로 Offset 만큼 좌표를 이동한다.

Kind: instance method of [GeometryCollection](#)

Overrides: [move](#)

Returns: [Geometry](#) - 이동된 공간 기하 객체

Param Type Description  
offsetX Number 가로 Offset  
offsetY Number 세로 Offset

geometryCollection.moveCentroid(중심)

주어진 중심좌표로 공간기하객체를 이동한다.

Kind: instance method of [GeometryCollection](#)

Param Type Description  
중심 [Coordinate](#) 좌표

geometryCollection.resize(upper, lower, left, right) ⇒ [Geometry](#)

상, 하, 좌, 우 외곽선을 이동하여 Envelope 을 리사이즈 한다.

Kind: instance method of [GeometryCollection](#)

Overrides: [resize](#)

Returns: [Geometry](#) - 리사이즈된 공간 기하 객체

Param Type Description  
upper Number 상단 라인 이동 Offset(위 방향으로 +)  
lower Number 하단 라인 이동 Offset(아래 방향으로 +)  
left Number 좌측 라인 이동 Offset(좌측 방향으로 +)  
right Number 우측 라인 이동 Offset(우측 방향으로 +)

geometryCollection.resizeBox(width, height) ⇒ [Geometry](#)

중심좌표는 고정한 채 Bounding Box 의 width, height 를 리사이즈 한다.

Kind: instance method of [GeometryCollection](#)

Overrides: [resizeBox](#)

Returns: [Geometry](#) - 리사이즈된 공간 기하 객체

Param Type Description  
width Number 너비  
height Number 높이

```
geometryCollection.rotate(angle, origin) => Geometry
```

기준 좌표를 기준으로 주어진 각도 만큼 회전한다.

Kind: instance method of [GeometryCollection](#)

Overrides: [rotate](#)

Returns: [Geometry](#) - 회전된 공간 기하 객체

Param	Type	Description
angle	Number	회전 각도
origin	<a href="#">Coordinate</a>	기준 좌표(default: 중심좌표)

```
geometryCollection.fitToBoundary(envelope) => Geometry
```

주어진 Boundary 영역 안으로 공간 기하 객체를 적용한다.(이동 & 리사이즈)

Kind: instance method of [GeometryCollection](#)

Overrides: [fitToBoundary](#)

Returns: [Geometry](#) - 적용된 공간 기하 객체

Param	Type	Description
envelope	<a href="#">Envelope</a>	Envelope 영역

```
geometryCollection.convertCoordinate(coordinate) => Coordinate
```

파라미터가 [x, y] 형식의 좌표 Array 이면 OG.geometry.Coordinate 인스턴스를 new 하여 반환한다.

Kind: instance method of [GeometryCollection](#)

Param	Type	Description
coordinate <code>&lt;Number&gt;</code>	<code>Coordinate</code>   Array. <code>&lt;Number&gt;</code>	[x, y] 형식의 좌표 Array 또는 OG.geometry.Coordinate 인스턴스

`geometryCollection.distanceToLine(p, line) ⇒ Number`

포인트 P로부터 라인 AB의 거리를 계산한다.

Note: NON-ROBUST!

Kind: instance method of [GeometryCollection](#)

Returns: Number - 거리

Param	Type	Description
p	<code>Coordinate</code>   Array. <code>&lt;Number&gt;</code>	기준좌표
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 시작좌표, 끝좌표 Array

`geometryCollection.distanceLineToLine(line1, line2) ⇒ Number`

라인1로부터 라인2의 거리를 계산한다.

Note: NON-ROBUST!

Kind: instance method of [GeometryCollection](#)

Returns: Number - 거리

Param	Type	Description
line1	<a href="#">Array.&lt;Coordinate&gt;</a>	line1 라인 시작좌표, 끝좌표 Array
line2	<a href="#">Array.&lt;Coordinate&gt;</a>	line2 라인 시작좌표, 끝좌표 Array

`geometryCollection.intersectToLine(line) ⇒ Array.<Coordinate>`

기하도형이 주어진 라인과 교차하는 좌표들을 반환한다.

Kind: instance method of [GeometryCollection](#)

Param	Type	Description
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 시작좌표, 끝좌표 Array

```
geometryCollection.shortestIntersectToLine(line) => Array.<Coordinate>
```

기하도형이 주어진 라인과 교차하는 좌표중 시작좌표에 가장 가까운 좌표를 반환한다.

Kind: instance method of [GeometryCollection](#)

Param	Type	Description
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 시작좌표, 끝좌표 Array

```
geometryCollection.intersectLineToLine(line1, line2, extension) => Coordinate
```

라인1 과 라인2 의 교차점을 계산한다.

Kind: instance method of [GeometryCollection](#)

Returns: [Coordinate](#) - 교차점

Param	Type	Description
line1	<a href="#">Array.&lt;Coordinate&gt;</a>	line1 라인 시작좌표, 끝좌표 Array
line2	<a href="#">Array.&lt;Coordinate&gt;</a>	line2 라인 시작좌표, 끝좌표 Array
extension	boolean	라인을 연장하여 교차점을 계산하는 여부

```
geometryCollection.intersectCircleToLine(center, radius, from, to) => Array.<Coordinate>
```

주어진 원과 주어진 라인의 교차점을 계산한다.

Kind: instance method of [GeometryCollection](#)

Returns: [Array.<Coordinate>](#) - 교차점

Param	Type	Description
center	<a href="#">Coordinate</a>	중심점
radius	Number	반경
from	<a href="#">Coordinate</a>	라인 시작좌표
to	<a href="#">Coordinate</a>	라인 끝좌표

geometryCollection.intersectPointToLine(p, line) ⇒ [Coordinate](#)

포인트 P로부터 라인 AB 까지 수직인 가상선을 생각할때, 그 교차점을 계산한다.

Note: NON-ROBUST!

Kind: instance method of [GeometryCollection](#)

Returns: [Coordinate](#) - 교차점

Param	Type	Description
p	<a href="#">Coordinate</a>   <a href="#">Array.&lt;Number&gt;</a>	기준좌표
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 시작좌표, 끝좌표 Array

geometryCollection.getPercentageDistanceFromPoint(\_coordinate) ⇒ Object

주어진 좌표에 대해 공간기하객체 바운더리 대비 가로, 세로 위치 퍼센테이지 비율을 구한다.

Kind: instance method of [GeometryCollection](#)

Returns: Object - {px , py}

Param	Type	Description
_coordinate	<a href="#">Coordinate</a>	좌표

geometryCollection.isContainsPoint(\_coordinate) ⇒ boolean

공간기하객체가 주어진 좌표를 포함하는지를 반환한다.

Kind: instance method of [GeometryCollection](#)

Returns: boolean - true, false

Param	Type	Description
_coordinate	<a href="#">Coordinate</a>	좌표

geometryCollection.getPointFromPercentageDistance(pXpY) ⇒ [Coordinate](#)

공간기하객체에 대한 퍼센테이지 좌표의 실제 좌표를 구한다.

Kind: instance method of [GeometryCollection](#)

Returns: [Coordinate](#) - 실 좌표

Param	Type	Description
pXpY	Array	퍼센테이지 좌표

geometryCollection.getParallelLine(from, to, distance) ⇒ [Array.<Coordinate>](#)

주어진 선분과 일정 거리에 있는 평행한 선분을 반환한다.

Kind: instance method of [GeometryCollection](#)

Returns: [Array.<Coordinate>](#) - 평행선 시작좌표, 끝좌표 Array

Param	Type	Description
from	<a href="#">Coordinate</a>	라인 시작좌표
to	<a href="#">Coordinate</a>	라인 끝좌표
distance		

```
geometryCollection.getParallelPath(line, distance)
```

주어진 라인과 일정 거리에 있는 평행한 라인을 반환한다.

Kind: instance method of [GeometryCollection](#)

Param	Type	Description
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 좌표 Array
distance		

```
geometryCollection.reset()
```

저장된 boundary 를 클리어하여 새로 계산하도록 한다.

Kind: instance method of [GeometryCollection](#)

geometry.Line ≈ [PolyLine](#)

Kind: static class of [geometry](#)

Extends: [PolyLine](#)

Requires: module:OG.geometry.Coordinate, module:OG.geometry.Envelope,  
module:OG.geometry.Geometry

Author: [Seungpil Park \(mailto:spspark@uengine.org\)](mailto:Seungpil_Park@uengine.org)

- `.Line`  $\Leftarrow$  `PolyLine`
  - `new OG.geometry.Line(from, to)`
  - `.vertices : Array.<Coordinate>`
  - `.TYPE : Number`
  - `.IS_CLOSED : Boolean`
  - `.style : Style`
  - `.boundary : Envelope`
  - `.getVertices() \Rightarrow Array.<Coordinate>`
  - `.move(offsetX, offsetY) \Rightarrow Geometry`
  - `.resize(upper, lower, left, right) \Rightarrow Geometry`
  - `.rotate(angle, origin) \Rightarrow Geometry`
  - `.toString() \Rightarrow String`
  - `.angleBetweenPoints(prev, next) \Rightarrow Number`
  - `.isRightAngleBetweenPoints(prev, next) \Rightarrow Object`
  - `.angleBetweenThreePoints(prev, next) \Rightarrow Number`
  - `.isEqual(_geometry) \Rightarrow Boolean`
  - `.isContains(_geometry) \Rightarrow Boolean`
  - `.isWithin(_geometry) \Rightarrow Boolean`
  - `.getBoundary() \Rightarrow Envelope`
  - `.getCentroid() \Rightarrow Coordinate`
  - `.minDistance(_coordinate) \Rightarrow Number`
  - `.distance(_geometry) \Rightarrow Number`
  - `.getLength() \Rightarrow Number`
  - `.moveCentroid(중심)`
  - `.resizeBox(width, height) \Rightarrow Geometry`

- [.fitToBoundary\(envelope\)](#) ⇒ [Geometry](#)
- [.convertCoordinate\(coordinate\)](#) ⇒ [Coordinate](#)
- [.distanceToLine\(p, line\)](#) ⇒ Number
- [.distanceLineToLine\(line1, line2\)](#) ⇒ Number
- [.intersectToLine\(line\)](#) ⇒ [Array.<Coordinate>](#)
- [.shortestIntersectToLine\(line\)](#) ⇒ [Array.<Coordinate>](#)
- [.intersectLineToLine\(line1, line2, extension\)](#) ⇒ [Coordinate](#)
- [.intersectCircleToLine\(center, radius, from, to\)](#) ⇒ [Array.<Coordinate>](#)
- [.intersectPointToLine\(p, line\)](#) ⇒ [Coordinate](#)
- [.getPercentageDistanceFromPoint\(\\_coordinate\)](#) ⇒ Object
- [.isContainsPoint\(\\_coordinate\)](#) ⇒ boolean
- [.getPointFromPercentageDistance\(pXpY\)](#) ⇒ [Coordinate](#)
- [.getParallelLine\(from, to, distance\)](#) ⇒ [Array.<Coordinate>](#)
- [.getParallelPath\(line, distance\)](#)
- [.reset\(\)](#)

new OG.geometry.Line(from, to)

Line 공간 기하 객체(Spatial Geometry Object)

Param	Type	Description
from	<a href="#">Coordinate</a>	라인 시작 좌표값
to	<a href="#">Coordinate</a>	라인 끝 좌표값

Example

```
var geom = new OG.geometry.Line([20, 5], [30, 15]);
```

line.vertices : [Array.<Coordinate>](#)

Line Vertex 좌표 Array

Kind: instance property of [Line](#)

line.TYPE : Number

공간 기하 객체 타입

Kind: instance property of [Line](#)

Overrides: [TYPE](#)

line.IS\_CLOSED : Boolean

닫힌 기하 객체 인지 여부

Kind: instance property of [Line](#)

line.style : [Style](#)

스타일 속성

Kind: instance property of [Line](#)

Overrides: [style](#)

line.boundary : [Envelope](#)

공간기하 객체를 포함하는 사각형의 Boundary 영역

Kind: instance property of [Line](#)

line.getVertices() ⇒ [Array.<Coordinate>](#)

공간기하 객체의 모든 꼭지점을 반환한다.

Kind: instance method of [Line](#)

Returns: [Array.<Coordinate>](#) - 꼭지점 좌표 Array

line.move(offsetX, offsetY) ⇒ [Geometry](#)

가로, 세로 Offset 만큼 좌표를 이동한다.

Kind: instance method of [Line](#)

Returns: [Geometry](#) - 이동된 공간 기하 객체

Param	Type	Description
offsetX	Number	가로 Offset
offsetY	Number	세로 Offset

line.resize(upper, lower, left, right) ⇒ [Geometry](#)

상, 하, 좌, 우 외곽선을 이동하여 Envelope 을 리사이즈 한다.

Kind: instance method of [Line](#)

Returns: [Geometry](#) - 리사이즈된 공간 기하 객체

Param	Type	Description
upper	Number	상단 라인 이동 Offset(위 방향으로 +)
lower	Number	하단 라인 이동 Offset(아래 방향으로 +)
left	Number	좌측 라인 이동 Offset(좌측 방향으로 +)
right	Number	우측 라인 이동 Offset(우측 방향으로 +)

`line.rotate(angle, origin) ⇒ Geometry`

기준 좌표를 기준으로 주어진 각도 만큼 회전한다.

Kind: instance method of [Line](#)

Returns: [Geometry](#) - 회전된 공간 기하 객체

Param	Type	Description
angle	Number	회전 각도
origin	<a href="#">Coordinate</a>	기준 좌표(default:중심좌표)

`line.toString() ⇒ String`

객체 프라퍼티 정보를 JSON 스트링으로 반환한다.

Kind: instance method of [Line](#)

Overrides: [toString](#)

Returns: String - 프라퍼티 정보

```
line.angleBetweenPoints(prev, next) => Number
```

공간기하객체의 두 꼭지점 사이에 가상의 선을 그렸을때, 그 기울기를 구한다.

Kind: instance method of [Line](#)

Returns: Number - 기울기

Param Type Description

prev [Coordinate](#) 꼭지점 1

next [Coordinate](#) 꼭지점 2

```
line.isRightAngleBetweenPoints(prev, next) => Object
```

공간기하객체의 두 꼭지점 사이의 기울기가 수평또는 수직인지 판별한다.

Kind: instance method of [Line](#)

Returns: Object - {flag : true or false, type: horizontal or vertical or none}

Param Type Description

prev [Coordinate](#) 꼭지점 1

next [Coordinate](#) 꼭지점 2

```
line.angleBetweenThreePoints(prev, next) => Number
```

공간기하객체의 세 꼭지점 사이의 각도 중 작은 각도를 반환한다.

Kind: instance method of [Line](#)

Returns: Number - 기울기

Param Type Description

prev [Coordinate](#) 꼭지점 1

next [Coordinate](#) 꼭지점 2

```
line.isEquals(_geometry) => Boolean
```

주어진 Geometry 객체와 같은지 비교한다.

Kind: instance method of [Line](#)

Returns: Boolean - true:같음, false:다름

Param Type Description

\_geometry [Geometry](#) Geometry 객체

```
line.isContains(_geometry) => Boolean
```

주어진 공간기하객체를 포함하는지 비교한다.

Kind: instance method of [Line](#)

Returns: Boolean - 포함하면 true

Param Type Description

\_geometry [Geometry](#) Geometry 객체

```
line.isWithin(_geometry) => Boolean
```

주어진 공간기하객체에 포함되는지 비교한다.

Kind: instance method of [Line](#)

Returns: Boolean - 포함되면 true

Param	Type	Description
_geometry	<a href="#">Geometry</a>	Geometry 객체

line.getBoundary() ⇒ [Envelope](#)

공간기하객체를 포함하는 사각형의 Boundary 영역을 반환한다.

Kind: instance method of [Line](#)

Returns: [Envelope](#) - Envelope 영역

line.getCentroid() ⇒ [Coordinate](#)

공간기하객체의 중심좌표를 반환한다.

Kind: instance method of [Line](#)

Returns: [Coordinate](#) - 중심좌표

line.minDistance(\_coordinate) ⇒ Number

주어진 좌표와의 최단거리를 반환한다.

Kind: instance method of [Line](#)

Returns: Number - 최단거리

Param	Type	Description
_coordinate	<a href="#">Coordinate</a>	좌표

```
line.distance(_geometry) => Number
```

주어진 공간기하 객체와의 중심점 간의 거리를 반환한다.

Kind: instance method of [Line](#)

Returns: Number - 거리

Param	Type	Description
_geometry	<a href="#">Geometry</a>	공간 기하 객체

```
line.getLength() => Number
```

공간기하 객체의 길이를 반환한다.

Kind: instance method of [Line](#)

Returns: Number - 길이

```
line.moveCentroid(중심)
```

주어진 중심좌표로 공간기하 객체를 이동한다.

Kind: instance method of [Line](#)

Param	Type	Description
중심	<a href="#">Coordinate</a>	좌표

```
line.resizeBox(width, height) => Geometry
```

중심좌표는 고정한 채 Bounding Box 의 width, height 를 리사이즈 한다.

Kind: instance method of [Line](#)

Returns: [Geometry](#) - 리사이즈된 공간 기하 객체

Param Type Description

width Number 너비

height Number 높이

line.fitToBoundary(envelope) ⇒ [Geometry](#)

주어진 Boundary 영역 안으로 공간 기하 객체를 적용한다.(이동 & 리사이즈)

Kind: instance method of [Line](#)

Returns: [Geometry](#) - 적용된 공간 기하 객체

Param Type Description

envelope [Envelope](#) Envelope 영역

line.convertCoordinate(coordinate) ⇒ [Coordinate](#)

파라미터가 [x, y] 형식의 좌표 Array 이면 OG.geometry.Coordinate 인스턴스를 new 하여 반환한다.

Kind: instance method of [Line](#)

Param	Type	Description
coordinate	<a href="#">Coordinate</a>   Array. <Number>	[x, y] 형식의 좌표 Array 또는 OG.geometry.Coordinate 인스턴스

```
line.distanceToLine(p, line) => Number
```

포인트 P로부터 라인 AB의 거리를 계산한다.

Note: NON-ROBUST!

Kind: instance method of [Line](#)

Returns: Number - 거리

Param	Type	Description
p	<a href="#">Coordinate</a>   Array.<Number>	기준좌표
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 시작좌표, 끝좌표 Array

```
line.distanceLineToLine(line1, line2) => Number
```

라인1로부터 라인2의 거리를 계산한다.

Note: NON-ROBUST!

Kind: instance method of [Line](#)

Returns: Number - 거리

Param	Type	Description
line1	<a href="#">Array.&lt;Coordinate&gt;</a>	line1 라인 시작좌표, 끝좌표 Array
line2	<a href="#">Array.&lt;Coordinate&gt;</a>	line2 라인 시작좌표, 끝좌표 Array

```
line.intersectToLine(line) => Array.<Coordinate>
```

기하도형이 주어진 라인과 교차하는 좌표들을 반환한다.

Kind: instance method of [Line](#)

Param	Type	Description
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 시작좌표, 끝좌표 Array

line.shortestIntersectToLine(line) ⇒ [Array.<Coordinate>](#)

기하도형이 주어진 라인과 교차하는 좌표중 시작좌표에 가장 가까운 좌표를 반환한다.

Kind: instance method of [Line](#)

Param	Type	Description
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 시작좌표, 끝좌표 Array

line.intersectLineToLine(line1, line2, extension) ⇒ [Coordinate](#)

라인1 과 라인2 의 교차점을 계산한다.

Kind: instance method of [Line](#)

Returns: [Coordinate](#) - 교차점

Param	Type	Description
line1	<a href="#">Array.&lt;Coordinate&gt;</a>	line1 라인 시작좌표, 끝좌표 Array
line2	<a href="#">Array.&lt;Coordinate&gt;</a>	line2 라인 시작좌표, 끝좌표 Array
extension	boolean	라인을 연장하여 교차점을 계산하는 여부

line.intersectCircleToLine(center, radius, from, to) ⇒ [Array.<Coordinate>](#)

주어진 원과 주어진 라인의 교차점을 계산한다.

Kind: instance method of [Line](#)

Returns: [Array.<Coordinate>](#) - 교차점

Param	Type	Description
center	<a href="#">Coordinate</a>	중심점
radius	Number	반경
from	<a href="#">Coordinate</a>	라인 시작좌표
to	<a href="#">Coordinate</a>	라인 끝좌표

```
line.intersectPointToLine(p, line) => Coordinate
```

포인트 P로부터 라인 AB 까지 수직인 가상선을 생각할때, 그 교차점을 계산한다.

Note: NON-ROBUST!

Kind: instance method of [Line](#)

Returns: [Coordinate](#) - 교차점

Param	Type	Description
p	<a href="#">Coordinate</a>   Array.<Number>	기준좌표
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 시작좌표, 끝좌표 Array

```
line.getPercentageDistanceFromPoint(_coordinate) => Object
```

주어진 좌표에 대해 공간기하객체 바운더리 대비 가로, 세로 위치 퍼센테이지 비율을 구한다.

Kind: instance method of [Line](#)

Returns: Object - {px , py}

Param	Type	Description
_coordinate	<a href="#">Coordinate</a>	좌표

```
line.isContainsPoint(_coordinate) => boolean
```

공간기하객체가 주어진 좌표를 포함하는지를 반환한다.

Kind: instance method of [Line](#)

Returns: boolean - true, false

Param	Type	Description
_coordinate	<a href="#">Coordinate</a>	좌표

line.getPointFromPercentageDistance(pXpY) ⇒ [Coordinate](#)

공간기하객체에 대한 퍼센테이지 좌표의 실제 좌표를 구한다.

Kind: instance method of [Line](#)

Returns: [Coordinate](#) - 실 좌표

Param	Type	Description
pXpY	Array	퍼센테이지 좌표

line.getParallelLine(from, to, distance) ⇒ [Array.<Coordinate>](#)

주어진 선분과 일정 거리에 있는 평행한 선분을 반환한다.

Kind: instance method of [Line](#)

Returns: [Array.<Coordinate>](#) - 평행선 시작좌표, 끝좌표 Array

Param	Type	Description
from	<a href="#">Coordinate</a>	라인 시작좌표
to	<a href="#">Coordinate</a>	라인 끝좌표
distance		

```
line.getParallelPath(line, distance)
```

주어진 라인과 일정 거리에 있는 평행한 라인을 반환한다.

Kind: instance method of [Line](#)

Param	Type	Description
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 좌표 Array
distance		

```
line.reset()
```

저장된 boundary 를 클리어하여 새로 계산하도록 한다.

Kind: instance method of [Line](#)

```
geometry.Point <= Geometry
```

Kind: static class of [geometry](#)

Extends: [Geometry](#)

Requires: module:OG.geometry.Coordinate, module:OG.geometry.Envelope,  
module:OG.geometry.Geometry

Author: [Seungpil Park \(mailto:sppark@uengine.org\)](mailto:Seungpil.Park@sppark@uengine.org)

- [.Point](#) <= [Geometry](#)
  - [new OG.geometry.Point\(coordinate\)](#)
  - [.coordinate](#) : [Coordinate](#)
  - [.vertices](#) : [Array.<Coordinate>](#)

- [.TYPE](#) : Number
- [.IS\\_CLOSED](#) : Boolean
- [.style](#) : [Style](#)
- [.boundary](#) : [Envelope](#)
- [.toString\(\)](#) ⇒ String
- [.equals\(\\_geometry\)](#) ⇒ Boolean
- [.contains\(\\_geometry\)](#) ⇒ Boolean
- [.within\(\\_geometry\)](#) ⇒ Boolean
- [.getBoundary\(\)](#) ⇒ [Envelope](#)
- [.getCentroid\(\)](#) ⇒ [Coordinate](#)
- [.getVertices\(\)](#) ⇒ [Array.<Coordinate>](#)
- [.minDistance\(\\_coordinate\)](#) ⇒ Number
- [.distance\(\\_geometry\)](#) ⇒ Number
- [.length\(\)](#) ⇒ Number
- [.move\(offsetX, offsetY\)](#) ⇒ [Geometry](#)
- [.moveCentroid\(중심\)](#)
- [.resize\(upper, lower, left, right\)](#) ⇒ [Geometry](#)
- [.resizeBox\(width, height\)](#) ⇒ [Geometry](#)
- [.rotate\(angle, origin\)](#) ⇒ [Geometry](#)
- [.fitToBoundary\(envelope\)](#) ⇒ [Geometry](#)
- [.convertCoordinate\(coordinate\)](#) ⇒ [Coordinate](#)
- [.distanceToLine\(p, line\)](#) ⇒ Number
- [.distanceLineToLine\(line1, line2\)](#) ⇒ Number
- [.intersectToLine\(line\)](#) ⇒ [Array.<Coordinate>](#)
- [.shortestIntersectToLine\(line\)](#) ⇒ [Array.<Coordinate>](#)
- [.intersectLineToLine\(line1, line2, extension\)](#) ⇒ [Coordinate](#)

- `.intersectCircleToLine(center, radius, from, to) ⇒ Array.<Coordinate>`
- `.intersectPointToLine(p, line) ⇒ Coordinate`
- `.getPercentageDistanceFromPoint(_coordinate) ⇒ Object`
- `.isContainsPoint(_coordinate) ⇒ boolean`
- `.getPointFromPercentageDistance(pXpY) ⇒ Coordinate`
- `.getParallelLine(from, to, distance) ⇒ Array.<Coordinate>`
- `.getParallelPath(line, distance)`
- `.reset()`

```
new OG.geometry.Point(coordinate)
```

Point 공간 기하 객체(Spatial Geometry Object)

Param	Type	Description
coordinate	<a href="#">Coordinate</a>	좌표값

Example

```
var geom = new OG.geometry.Point([20, 5]);
```

```
point.coordinate : Coordinate
```

좌표값

Kind: instance property of [Point](#)

point.vertices : [Array.<Coordinate>](#)

Line Vertex 좌표 Array

Kind: instance property of [Point](#)

point.TYPE : Number

공간 기하 객체 타입

Kind: instance property of [Point](#)

Overrides: [TYPE](#)

point.IS\_CLOSED : Boolean

닫힌 기하 객체 인지 여부

Kind: instance property of [Point](#)

point.style : [Style](#)

스타일 속성

Kind: instance property of [Point](#)

Overrides: [style](#)

`point.boundary` : [Envelope](#)

공간기하객체를 포함하는 사각형의 Boundary 영역

Kind: instance property of [Point](#)

Overrides: [boundary](#)

`point.toString()` ⇒ String

객체 프라퍼티 정보를 JSON 스트링으로 반환한다.

Kind: instance method of [Point](#)

Returns: String - 프라퍼티 정보

`point.isEquals(_geometry)` ⇒ Boolean

주어진 Geometry 객체와 같은지 비교한다.

Kind: instance method of [Point](#)

Returns: Boolean - true:같음, false:다름

Param	Type	Description
<code>_geometry</code>	<a href="#">Geometry</a>	Geometry 객체

`point.isContains(_geometry)` ⇒ Boolean

주어진 공간기하객체를 포함하는지 비교한다.

Kind: instance method of [Point](#)

Returns: Boolean - 포함하면 true

Param	Type	Description
_geometry	<a href="#">Geometry</a>	Geometry 객체

point.isWithin(\_geometry) ⇒ Boolean

주어진 공간기하객체에 포함되는지 비교한다.

Kind: instance method of [Point](#)

Returns: Boolean - 포함되면 true

Param	Type	Description
_geometry	<a href="#">Geometry</a>	Geometry 객체

point.getBoundary() ⇒ [Envelope](#)

공간기하객체를 포함하는 사각형의 Boundary 영역을 반환한다.

Kind: instance method of [Point](#)

Returns: [Envelope](#) - Envelope 영역

point.getCentroid() ⇒ [Coordinate](#)

공간기하객체의 중심좌표를 반환한다.

Kind: instance method of [Point](#)

Returns: [Coordinate](#) - 중심좌표

point.getVertices() ⇒ [Array.<Coordinate>](#)

공간기하객체의 모든 꼭지점을 반환한다.

Kind: instance method of [Point](#)

Overrides: [getVertices](#)

Returns: [Array.<Coordinate>](#) - 꼭지점 좌표 Array

point.minDistance(\_coordinate) ⇒ Number

주어진 좌표와의 최단거리를 반환한다.

Kind: instance method of [Point](#)

Returns: Number - 최단거리

Param	Type	Description
_coordinate	<a href="#">Coordinate</a>	좌표

point.distance(\_geometry) ⇒ Number

주어진 공간기하객체와의 중심점 간의 거리를 반환한다.

Kind: instance method of [Point](#)

Returns: Number - 거리

Param	Type	Description
_geometry	<a href="#">Geometry</a>	공간 기하 객체

point.getLength() ⇒ Number

공간기하객체의 길이를 반환한다.

Kind: instance method of [Point](#)

Returns: Number - 길이

point.move(offsetX, offsetY) ⇒ [Geometry](#)

가로, 세로 Offset 만큼 좌표를 이동한다.

Kind: instance method of [Point](#)

Overrides: [move](#)

Returns: [Geometry](#) - 이동된 공간 기하 객체

Param	Type	Description
offsetX	Number	가로 Offset
offsetY	Number	세로 Offset

point.moveCentroid(중심)

주어진 중심좌표로 공간기하객체를 이동한다.

Kind: instance method of [Point](#)

Overrides: [moveCentroid](#)

Param Type Description

중심 [Coordinate](#) 좌표

point.resize(upper, lower, left, right) ⇒ [Geometry](#)

상, 하, 좌, 우 외곽선을 이동하여 Envelope 을 리사이즈 한다.

Kind: instance method of [Point](#)

Overrides: [resize](#)

Returns: [Geometry](#) - 리사이즈된 공간 기하 객체

Param Type Description

upper Number 상단 라인 이동 Offset(위 방향으로 +)

lower Number 하단 라인 이동 Offset(아래 방향으로 +)

left Number 좌측 라인 이동 Offset(좌측 방향으로 +)

right Number 우측 라인 이동 Offset(우측 방향으로 +)

point.resizeBox(width, height) ⇒ [Geometry](#)

중심좌표는 고정한 채 Bounding Box 의 width, height 를 리사이즈 한다.

Kind: instance method of [Point](#)

Overrides: [resizeBox](#)

Returns: [Geometry](#) - 리사이즈된 공간 기하 객체

Param Type Description

width Number 너비

height Number 높이

point.rotate(angle, origin) ⇒ [Geometry](#)

기준 좌표를 기준으로 주어진 각도 만큼 회전한다.

Kind: instance method of [Point](#)

Overrides: [rotate](#)

Returns: [Geometry](#) - 회전된 공간 기하 객체

Param	Type	Description
angle	Number	회전 각도
origin	<a href="#">Coordinate</a>	기준 좌표(default: 중심좌표)

`point.fitToBoundary(envelope) ⇒ Geometry`

주어진 Boundary 영역 안으로 공간 기하 객체를 적용한다.(이동 & 리사이즈)

Kind: instance method of [Point](#)

Overrides: [fitToBoundary](#)

Returns: [Geometry](#) - 적용된 공간 기하 객체

Param	Type	Description
envelope	<a href="#">Envelope</a>	Envelope 영역

`point.convertCoordinate(coordinate) ⇒ Coordinate`

파라미터가 [x, y] 형식의 좌표 Array 이면 OG.geometry.Coordinate 인스턴스를 new 하여 반환한다.

Kind: instance method of [Point](#)

Param	Type	Description
<a href="#">Coordinate</a>   Array.	[x, y]	형식의 좌표 Array 또는

```
coordinate <Number>
```

OG.geometry.Coordinate 인스턴스

```
point.distanceToLine(p, line) ⇒ Number
```

포인트 P로부터 라인 AB의 거리를 계산한다.

Note: NON-ROBUST!

Kind: instance method of [Point](#)

Returns: Number - 거리

Param	Type	Description
p	<a href="#">Coordinate</a>   Array.<Number>	기준좌표
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 시작좌표, 끝좌표 Array

```
point.distanceLineToLine(line1, line2) ⇒ Number
```

라인1로부터 라인2의 거리를 계산한다.

Note: NON-ROBUST!

Kind: instance method of [Point](#)

Returns: Number - 거리

Param	Type	Description
line1	<a href="#">Array.&lt;Coordinate&gt;</a>	line1 라인 시작좌표, 끝좌표 Array
line2	<a href="#">Array.&lt;Coordinate&gt;</a>	line2 라인 시작좌표, 끝좌표 Array

```
point.intersectToLine(line) ⇒ Array.<Coordinate>
```

기하도형이 주어진 라인과 교차하는 좌표들을 반환한다.

Kind: instance method of [Point](#)

Param	Type	Description
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 시작좌표, 끝좌표 Array

point.shortestIntersectToLine(line) ⇒ [Array.<Coordinate>](#)

기하도형이 주어진 라인과 교차하는 좌표중 시작좌표에 가장 가까운 좌표를 반환한다.

Kind: instance method of [Point](#)

Param	Type	Description
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 시작좌표, 끝좌표 Array

point.intersectLineToLine(line1, line2, extension) ⇒ [Coordinate](#)

라인1 과 라인2 의 교차점을 계산한다.

Kind: instance method of [Point](#)

Returns: [Coordinate](#) - 교차점

Param	Type	Description
line1	<a href="#">Array.&lt;Coordinate&gt;</a>	line1 라인 시작좌표, 끝좌표 Array
line2	<a href="#">Array.&lt;Coordinate&gt;</a>	line2 라인 시작좌표, 끝좌표 Array
extension	boolean	라인을 연장하여 교차점을 계산하는 여부

point.intersectCircleToLine(center, radius, from, to) ⇒ [Array.<Coordinate>](#)

주어진 원과 주어진 라인의 교차점을 계산한다.

Kind: instance method of [Point](#)

Returns: [Array.<Coordinate>](#) - 교차점

Param	Type	Description
center	<a href="#">Coordinate</a>	중심점
radius	Number	반경
from	<a href="#">Coordinate</a>	line 라인 시작좌표
to	<a href="#">Coordinate</a>	line 라인 끝좌표

```
point.intersectPointToLine(p, line) => Coordinate
```

포인트 P로부터 라인 AB 까지 수직인 가상선을 생각할때, 그 교차점을 계산한다.

Note: NON-ROBUST!

Kind: instance method of [Point](#)

Returns: [Coordinate](#) - 교차점

Param	Type	Description
p	<a href="#">Coordinate</a>   Array.<Number>	기준좌표
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 시작좌표, 끝좌표 Array

```
point.getPercentageDistanceFromPoint(_coordinate) => Object
```

주어진 좌표에 대해 공간기하객체 바운더리 대비 가로, 세로 위치 퍼센테이지 비율을 구한다.

Kind: instance method of [Point](#)

Returns: Object - {px , py}

Param	Type	Description
_coordinate	<a href="#">Coordinate</a>	좌표

```
point.isContainsPoint(_coordinate) => boolean
```

공간기하객체가 주어진 좌표를 포함하는지를 반환한다.

Kind: instance method of [Point](#)

Returns: boolean - true, false

Param	Type	Description
_coordinate	<a href="#">Coordinate</a>	좌표

```
point.getPointFromPercentageDistance(pXpY) => Coordinate
```

공간기하객체에 대한 퍼센테이지 좌표의 실제 좌표를 구한다.

Kind: instance method of [Point](#)

Returns: [Coordinate](#) - 실 좌표

Param	Type	Description
pXpY	Array	퍼센테이지 좌표

```
point.getParallelLine(from, to, distance) => Array.<Coordinate>
```

주어진 선분과 일정 거리에 있는 평행한 선분을 반환한다.

Kind: instance method of [Point](#)

Returns: [Array.<Coordinate>](#) - 평행선 시작좌표, 끝좌표 Array

Param	Type	Description
from	<a href="#">Coordinate</a>	라인 시작좌표

to        [Coordinate](#) 라인 끝좌표  
distance

```
point.getParallelPath(line, distance)
```

주어진 라인과 일정 거리에 있는 평행한 라인을 반환한다.

Kind: instance method of [Point](#)

Param	Type	Description
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 좌표 Array
distance		

```
point.reset()
```

저장된 boundary 를 클리어하여 새로 계산하도록 한다.

Kind: instance method of [Point](#)

[geometry.PolyLine](#) <= [Geometry](#)

Kind: static class of [geometry](#)

Extends: [Geometry](#)

Requires: module:OG.geometry.Coordinate, module:OG.geometry.Envelope,  
module:OG.geometry.Geometry

Author: [Seungpil Park \(mailto:sppark@uengine.org\)](mailto:Seungpil.Park@sppark@uengine.org)

- [.PolyLine](#) <= [Geometry](#)

- [`new OG.geometry.PolyLine\(vertices\)`](#)
- [`.vertices : Array.<Coordinate>`](#)
- [`.TYPE : Number`](#)
- [`.IS\_CLOSED : Boolean`](#)
- [`.style : Style`](#)
- [`.boundary : Envelope`](#)
- [`.toString\(\) => String`](#)
- [`.angleBetweenPoints\(prev, next\) => Number`](#)
- [`.isRightAngleBetweenPoints\(prev, next\) => Object`](#)
- [`.angleBetweenThreePoints\(prev, next\) => Number`](#)
- [`.isEqual\(\_geometry\) => Boolean`](#)
- [`.isContains\(\_geometry\) => Boolean`](#)
- [`.isWithin\(\_geometry\) => Boolean`](#)
- [`.getBoundary\(\) => Envelope`](#)
- [`.getCentroid\(\) => Coordinate`](#)
- [`.getVertices\(\) => Array.<Coordinate>`](#)
- [`.minDistance\(\_coordinate\) => Number`](#)
- [`.distance\( geometry \) => Number`](#)
- [`.getLength\(\) => Number`](#)
- [`.move\(offsetX, offsetY\) => Geometry`](#)
- [`.moveCentroid\(중심\)`](#)
- [`.resize\(upper, lower, left, right\) => Geometry`](#)
- [`.resizeBox\(width, height\) => Geometry`](#)
- [`.rotate\(angle, origin\) => Geometry`](#)
- [`.fitToBoundary\(envelope\) => Geometry`](#)
- [`.convertCoordinate\(coordinate\) => Coordinate`](#)

- [.distanceToLine\(p, line\)](#) ⇒ Number
- [.distanceLineToLine\(line1, line2\)](#) ⇒ Number
- [.intersectToLine\(line\)](#) ⇒ [Array.<Coordinate>](#)
- [.shortestIntersectToLine\(line\)](#) ⇒ [Array.<Coordinate>](#)
- [.intersectLineToLine\(line1, line2, extension\)](#) ⇒ [Coordinate](#)
- [.intersectCircleToLine\(center, radius, from, to\)](#) ⇒ [Array.<Coordinate>](#)
- [.intersectPointToLine\(p, line\)](#) ⇒ [Coordinate](#)
- [.getPercentageDistanceFromPoint\(\\_coordinate\)](#) ⇒ Object
- [.isContainsPoint\(\\_coordinate\)](#) ⇒ boolean
- [.getPointFromPercentageDistance\(pXpY\)](#) ⇒ [Coordinate](#)
- [.getParallelLine\(from, to, distance\)](#) ⇒ [Array.<Coordinate>](#)
- [.getParallelPath\(line, distance\)](#)
- [.reset\(\)](#)

```
new OG.geometry.PolyLine(vertices)
```

PolyLine 공간 기하 객체(Spatial Geometry Object)

Param	Type	Description
vertices	<a href="#"><u>Array.&lt;Coordinate&gt;</u></a>	Line Vertex 좌표 Array

Example

```
var geom = new OG.geometry.PolyLine([[20, 5], [30, 15], [40, 25], [50, 15]]);
```

`polyLine.vertices : Array.<Coordinate>`

Line Vertex 좌표 Array

Kind: instance property of [PolyLine](#)

`polyLine.TYPE : Number`

공간 기하 객체 타입

Kind: instance property of [PolyLine](#)

Overrides: [TYPE](#)

`polyLine.IS_CLOSED : Boolean`

닫힌 기하 객체 인지 여부

Kind: instance property of [PolyLine](#)

`polyLine.style : Style`

스타일 속성

Kind: instance property of [PolyLine](#)

Overrides: [style](#)

```
polyLine.boundary : Envelope
```

공간기하객체를 포함하는 사각형의 Boundary 영역

Kind: instance property of [PolyLine](#)

```
polyLine.toString() => String
```

객체 프라퍼티 정보를 JSON 스트링으로 반환한다.

Kind: instance method of [PolyLine](#)

Returns: String - 프라퍼티 정보

```
polyLine.angleBetweenPoints(prev, next) => Number
```

공간기하객체의 두 꼭지점 사이에 가상의 선을 그렸을때, 그 기울기를 구한다.

Kind: instance method of [PolyLine](#)

Returns: Number - 기울기

Param Type Description

prev [Coordinate](#) 꼭지점 1

next [Coordinate](#) 꼭지점 2

```
polyLine.isRightAngleBetweenPoints(prev, next) => Object
```

공간기하객체의 두 꼭지점 사이의 기울기가 수평또는 수직인지 판별한다.

Kind: instance method of [PolyLine](#)

Returns: Object - {flag : true or false, type: horizontal or vertical or none}

Param Type Description

prev [Coordinate](#) 꼭지점 1

next [Coordinate](#) 꼭지점 2

`polyLine.angleBetweenThreePoints(prev, next) ⇒ Number`

공간기하객체의 세 꼭지점 사이의 각도 중 작은 각도를 반환한다.

Kind: instance method of [PolyLine](#)

Returns: Number - 기울기

Param Type Description

prev [Coordinate](#) 꼭지점 1

next [Coordinate](#) 꼭지점 2

`polyLine.isEquals(_geometry) ⇒ Boolean`

주어진 Geometry 객체와 같은지 비교한다.

Kind: instance method of [PolyLine](#)

Returns: Boolean - true:같음, false:다름

Param Type Description

\_geometry [Geometry](#) Geometry 객체

`polyLine.isContains(_geometry) ⇒ Boolean`

주어진 공간기하객체를 포함하는지 비교한다.

Kind: instance method of [PolyLine](#)

Returns: Boolean - 포함하면 true

Param	Type	Description
_geometry	<a href="#">Geometry</a>	Geometry 객체

```
polyLine.isWithin(_geometry) ⇒ Boolean
```

주어진 공간기하객체에 포함되는지 비교한다.

Kind: instance method of [PolyLine](#)

Returns: Boolean - 포함되면 true

Param	Type	Description
_geometry	<a href="#">Geometry</a>	Geometry 객체

```
polyLine.getBoundary() ⇒ Envelope
```

공간기하객체를 포함하는 사각형의 Boundary 영역을 반환한다.

Kind: instance method of [PolyLine](#)

Returns: [Envelope](#) - Envelope 영역

```
polyLine.getCentroid() ⇒ Coordinate
```

공간기하객체의 중심좌표를 반환한다.

Kind: instance method of [PolyLine](#)

Returns: [Coordinate](#) - 중심좌표

`polyLine.getVertices() ⇒ Array.<Coordinate>`

공간기하객체의 모든 꼭지점을 반환한다.

Kind: instance method of [PolyLine](#)

Overrides: [getVertices](#)

Returns: [Array.<Coordinate>](#) - 꼭지점 좌표 Array

`polyLine.minDistance(_coordinate) ⇒ Number`

주어진 좌표와의 최단거리를 반환한다.

Kind: instance method of [PolyLine](#)

Returns: Number - 최단거리

Param	Type	Description
<code>_coordinate</code>	<a href="#">Coordinate</a>	좌표

`polyLine.distance(_geometry) ⇒ Number`

주어진 공간기하객체와의 중심점 간의 거리를 반환한다.

Kind: instance method of [PolyLine](#)

Returns: Number - 거리

Param	Type	Description
_geometry	<a href="#">Geometry</a>	공간 기하 객체

`polyLine.getLength() ⇒ Number`

공간기하객체의 길이를 반환한다.

Kind: instance method of [PolyLine](#)

Returns: Number - 길이

`polyLine.move(offsetX, offsetY) ⇒ Geometry`

가로, 세로 Offset 만큼 좌표를 이동한다.

Kind: instance method of [PolyLine](#)

Overrides: [move](#)

Returns: [Geometry](#) - 이동된 공간 기하 객체

Param	Type	Description
offsetX	Number	가로 Offset
offsetY	Number	세로 Offset

`polyLine.moveCentroid(중심)`

주어진 중심좌표로 공간기하객체를 이동한다.

Kind: instance method of [PolyLine](#)

Param	Type	Description

중심 [Coordinate](#) 좌표

`polyLine.resize(upper, lower, left, right) ⇒ Geometry`

상, 하, 좌, 우 외곽선을 이동하여 Envelope 을 리사이즈 한다.

Kind: instance method of [PolyLine](#)

Overrides: [resize](#)

Returns: [Geometry](#) - 리사이즈된 공간 기하 객체

Param	Type	Description
upper	Number	상단 라인 이동 Offset(위 방향으로 +)
lower	Number	하단 라인 이동 Offset(아래 방향으로 +)
left	Number	좌측 라인 이동 Offset(좌측 방향으로 +)
right	Number	우측 라인 이동 Offset(우측 방향으로 +)

`polyLine.resizeBox(width, height) ⇒ Geometry`

중심좌표는 고정한 채 Bounding Box 의 width, height 를 리사이즈 한다.

Kind: instance method of [PolyLine](#)

Returns: [Geometry](#) - 리사이즈된 공간 기하 객체

Param	Type	Description
width	Number	너비
height	Number	높이

`polyLine.rotate(angle, origin) ⇒ Geometry`

기준 좌표를 기준으로 주어진 각도 만큼 회전한다.

Kind: instance method of [PolyLine](#)

Overrides: [rotate](#)

Returns: [Geometry](#) - 회전된 공간 기하 객체

Param	Type	Description
angle	Number	회전 각도
origin	<a href="#">Coordinate</a>	기준 좌표(default: 중심좌표)

`polyLine.fitToBoundary(envelope) ⇒ Geometry`

주어진 Boundary 영역 안으로 공간 기하 객체를 적용한다.(이동 & 리사이즈)

Kind: instance method of [PolyLine](#)

Returns: [Geometry](#) - 적용된 공간 기하 객체

Param	Type	Description
envelope	<a href="#">Envelope</a>	Envelope 영역

`polyLine.convertCoordinate(coordinate) ⇒ Coordinate`

파라미터가 [x, y] 형식의 좌표 Array 이면 OG.geometry.Coordinate 인스턴스를 new 하여 반환한다.

Kind: instance method of [PolyLine](#)

Param	Type	Description
coordinate	<a href="#">Coordinate</a>   Array. <a href="#">Number</a>	[x, y] 형식의 좌표 Array 또는 OG.geometry.Coordinate 인스턴스

`polyLine.distanceToLine(p, line) ⇒ Number`

포인트 P로부터 라인 AB의 거리를 계산한다.

Note: NON-ROBUST!

Kind: instance method of [PolyLine](#)

Returns: Number - 거리

Param	Type	Description
p	<a href="#">Coordinate</a>   Array.<Number>	기준좌표
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 시작좌표, 끝좌표 Array

```
polyLine.distanceLineToLine(line1, line2) => Number
```

라인1로부터 라인2의 거리를 계산한다.

Note: NON-ROBUST!

Kind: instance method of [PolyLine](#)

Returns: Number - 거리

Param	Type	Description
line1	<a href="#">Array.&lt;Coordinate&gt;</a>	line1 라인 시작좌표, 끝좌표 Array
line2	<a href="#">Array.&lt;Coordinate&gt;</a>	line2 라인 시작좌표, 끝좌표 Array

```
polyLine.intersectToLine(line) => Array.<Coordinate>
```

기하도형이 주어진 라인과 교차하는 좌표들을 반환한다.

Kind: instance method of [PolyLine](#)

Param	Type	Description
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 시작좌표, 끝좌표 Array

```
polyLine.shortestIntersectToLine(line) ⇒ Array.<Coordinate>
```

기하도형이 주어진 라인과 교차하는 좌표중 시작좌표에 가장 가까운 좌표를 반환한다.

Kind: instance method of [PolyLine](#)

Param	Type	Description
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 시작좌표, 끝좌표 Array

```
polyLine.intersectLineToLine(line1, line2, extension) ⇒ Coordinate
```

라인1 과 라인2 의 교차점을 계산한다.

Kind: instance method of [PolyLine](#)

Returns: [Coordinate](#) - 교차점

Param	Type	Description
line1	<a href="#">Array.&lt;Coordinate&gt;</a>	line1 라인 시작좌표, 끝좌표 Array
line2	<a href="#">Array.&lt;Coordinate&gt;</a>	line2 라인 시작좌표, 끝좌표 Array
extension	boolean	라인을 연장하여 교차점을 계산하는 여부

```
polyLine.intersectCircleToLine(center, radius, from, to) ⇒ Array.<Coordinate>
```

주어진 원과 주어진 라인의 교차점을 계산한다.

Kind: instance method of [PolyLine](#)

Returns: [Array.<Coordinate>](#) - 교차점

Param	Type	Description
-------	------	-------------

```
center Coordinate 중심점  
radius Number 반경  
from Coordinate line 라인 시작좌표  
to Coordinate line 라인 끝좌표
```

```
polyLine.intersectPointToLine(p, line) ⇒ Coordinate
```

포인트 P로부터 라인 AB 까지 수직인 가상선을 생각할때, 그 교차점을 계산한다.

Note: NON-ROBUST!

Kind: instance method of [PolyLine](#)

Returns: [Coordinate](#) - 교차점

Param	Type	Description
p	<a href="#">Coordinate</a>   Array.<Number>	기준좌표
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 시작좌표, 끝좌표 Array

```
polyLine.getPercentageDistanceFromPoint(_coordinate) ⇒ Object
```

주어진 좌표에 대해 공간기하객체 바운더리 대비 가로, 세로 위치 퍼센테이지 비율을 구한다.

Kind: instance method of [PolyLine](#)

Returns: Object - {px , py}

Param	Type	Description
_coordinate	<a href="#">Coordinate</a>	좌표

```
polyLine.isContainsPoint(_coordinate) ⇒ boolean
```

공간기하객체가 주어진 좌표를 포함하는지를 반환한다.

Kind: instance method of [PolyLine](#)

Returns: boolean - true, false

Param	Type	Description
_coordinate	<a href="#">Coordinate</a>	좌표

`polyLine.getPointFromPercentageDistance(pXpY) ⇒ Coordinate`

공간기하객체에 대한 퍼센테이지 좌표의 실제 좌표를 구한다.

Kind: instance method of [PolyLine](#)

Returns: [Coordinate](#) - 실 좌표

Param	Type	Description
pXpY	Array	퍼센테이지 좌표

`polyLine.getParallelLine(from, to, distance) ⇒ Array.<Coordinate>`

주어진 선분과 일정 거리에 있는 평행한 선분을 반환한다.

Kind: instance method of [PolyLine](#)

Returns: [Array.<Coordinate>](#) - 평행선 시작좌표, 끝좌표 Array

Param	Type	Description
from	<a href="#">Coordinate</a>	라인 시작좌표
to	<a href="#">Coordinate</a>	라인 끝좌표
distance		

`polyLine.getParallelPath(line, distance)`

주어진 라인과 일정 거리에 있는 평행한 라인을 반환한다.

Kind: instance method of [PolyLine](#)

Param	Type	Description
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 좌표 Array
distance		

`polyLine.reset()`

저장된 boundary 를 클리어하여 새로 계산하도록 한다.

Kind: instance method of [PolyLine](#)

`geometry.Polygon`  $\leftarrow$  [PolyLine](#)

Kind: static class of [geometry](#)

Extends: [PolyLine](#)

Requires: module:OG.geometry.Coordinate, module:OG.geometry.Envelope, module:OG.geometry.Geometry

Author: [Seungpil Park \(mailto:sppark@uengine.org\)](mailto:Seungpil_Park (mailto:sppark@uengine.org))

- [.Polygon](#)  $\leftarrow$  [PolyLine](#)

- [new OG.geometry.Polygon\(vertices\)](#)
- [.vertices](#) : [Array.<Coordinate>](#)
- [.TYPE](#) : Number
- [.IS\\_CLOSED](#) : Boolean

- [.style](#) : [Style](#)
- [.boundary](#) : [Envelope](#)
- [.getVertices\(\)](#) ⇒ [Array.<Coordinate>](#)
- [.move\(offsetX, offsetY\)](#) ⇒ [Geometry](#)
- [.resize\(upper, lower, left, right\)](#) ⇒ [Geometry](#)
- [.rotate\(angle, origin\)](#) ⇒ [Geometry](#)
- [.toString\(\)](#) ⇒ [String](#)
- [.angleBetweenPoints\(prev, next\)](#) ⇒ [Number](#)
- [.isRightAngleBetweenPoints\(prev, next\)](#) ⇒ [Object](#)
- [.angleBetweenThreePoints\(prev, next\)](#) ⇒ [Number](#)
- [.isEqual\(\\_geometry\)](#) ⇒ [Boolean](#)
- [.isContains\(\\_geometry\)](#) ⇒ [Boolean](#)
- [.isWithin\(\\_geometry\)](#) ⇒ [Boolean](#)
- [.getBoundary\(\)](#) ⇒ [Envelope](#)
- [.getCentroid\(\)](#) ⇒ [Coordinate](#)
- [.minDistance\(\\_coordinate\)](#) ⇒ [Number](#)
- [.distance\(\\_geometry\)](#) ⇒ [Number](#)
- [.getLength\(\)](#) ⇒ [Number](#)
- [.moveCentroid\(중심\)](#)
- [.resizeBox\(width, height\)](#) ⇒ [Geometry](#)
- [.fitToBoundary\(envelope\)](#) ⇒ [Geometry](#)
- [.convertCoordinate\(coordinate\)](#) ⇒ [Coordinate](#)
- [.distanceToLine\(p, line\)](#) ⇒ [Number](#)
- [.distanceLineToLine\(line1, line2\)](#) ⇒ [Number](#)
- [.intersectToLine\(line\)](#) ⇒ [Array.<Coordinate>](#)
- [.shortestIntersectToLine\(line\)](#) ⇒ [Array.<Coordinate>](#)

- [.intersectLineToLine\(line1, line2, extension\)](#) ⇒ [Coordinate](#)
- [.intersectCircleToLine\(center, radius, from, to\)](#) ⇒ [Array.<Coordinate>](#)
- [.intersectPointToLine\(p, line\)](#) ⇒ [Coordinate](#)
- [.getPercentageDistanceFromPoint\(\\_coordinate\)](#) ⇒ Object
- [.isContainsPoint\(\\_coordinate\)](#) ⇒ boolean
- [.getPointFromPercentageDistance\(pXpY\)](#) ⇒ [Coordinate](#)
- [.getParallelLine\(from, to, distance\)](#) ⇒ [Array.<Coordinate>](#)
- [.getParallelPath\(line, distance\)](#)
- [.reset\(\)](#)

```
new OG.geometry.Polygon(vertices)
```

Polygon 공간 기하 객체(Spatial Geometry Object)

Param	Type	Description
vertices	<a href="#">Array.&lt;Coordinate&gt;</a>	Line Vertex 좌표 Array

Example

```
var geom = new OG.geometry.Polygon([[20, 5], [30, 15], [40, 25], [50, 15], [60, 5], [20, 5]]);
```

`polygon.vertices : Array.<Coordinate>`

Line Vertex 좌표 Array

Kind: instance property of [Polygon](#)

`polygon.TYPE : Number`

공간 기하 객체 타입

Kind: instance property of [Polygon](#)

Overrides: [TYPE](#)

`polygon.IS_CLOSED : Boolean`

닫힌 기하 객체 인지 여부

Kind: instance property of [Polygon](#)

Overrides: [IS\\_CLOSED](#)

`polygon.style : Style`

스타일 속성

Kind: instance property of [Polygon](#)

Overrides: [style](#)

`polygon.boundary : Envelope`

공간기하객체를 포함하는 사각형의 Boundary 영역

Kind: instance property of [Polygon](#)

`polygon.getVertices() ⇒ Array.<Coordinate>`

공간기하 객체의 모든 꼭지점을 반환한다.

Kind: instance method of [Polygon](#)

Returns: [Array.<Coordinate>](#) - 꼭지점 좌표 Array

`polygon.move(offsetX, offsetY) ⇒ Geometry`

가로, 세로 Offset 만큼 좌표를 이동한다.

Kind: instance method of [Polygon](#)

Returns: [Geometry](#) - 이동된 공간 기하 객체

Param	Type	Description
offsetX	Number	가로 Offset
offsetY	Number	세로 Offset

`polygon.resize(upper, lower, left, right) ⇒ Geometry`

상, 하, 좌, 우 외곽선을 이동하여 Envelope 을 리사이즈 한다.

Kind: instance method of [Polygon](#)

Returns: [Geometry](#) - 리사이즈된 공간 기하 객체

Param	Type	Description
-------	------	-------------

```
upper Number 상단 라인 이동 Offset(위 방향으로 +)
lower Number 하단 라인 이동 Offset(아래 방향으로 +)
left Number 좌측 라인 이동 Offset(좌측 방향으로 +)
right Number 우측 라인 이동 Offset(우측 방향으로 +)
```

```
polygon.rotate(angle, origin) ⇒ Geometry
```

기준 좌표를 기준으로 주어진 각도 만큼 회전한다.

Kind: instance method of [Polygon](#)

Returns: [Geometry](#) - 회전된 공간 기하 객체

Param	Type	Description
angle	Number	회전 각도
origin	<a href="#">Coordinate</a>	기준 좌표(default: 중심좌표)

```
polygon.toString() ⇒ String
```

객체 프라퍼티 정보를 JSON 스트링으로 반환한다.

Kind: instance method of [Polygon](#)

Returns: String - 프라퍼티 정보

```
polygon.angleBetweenPoints(prev, next) ⇒ Number
```

공간기하객체의 두 꼭지점 사이에 가상의 선을 그렸을때, 그 기울기를 구한다.

Kind: instance method of [Polygon](#)

Returns: Number - 기울기

Param	Type	Description
prev	<a href="#">Coordinate</a>	꼭지점 1
next	<a href="#">Coordinate</a>	꼭지점 2

```
polygon.isRightAngleBetweenPoints(prev, next) => Object
```

공간기하객체의 두 꼭지점 사이의 기울기가 수평또는 수직인지 판별한다.

Kind: instance method of [Polygon](#)

Returns: Object - {flag : true or false, type: horizontal or vertical or none}

Param	Type	Description
prev	<a href="#">Coordinate</a>	꼭지점 1
next	<a href="#">Coordinate</a>	꼭지점 2

```
polygon.angleBetweenThreePoints(prev, next) => Number
```

공간기하객체의 세 꼭지점 사이의 각도 중 작은 각도를 반환한다.

Kind: instance method of [Polygon](#)

Returns: Number - 기울기

Param	Type	Description
prev	<a href="#">Coordinate</a>	꼭지점 1
next	<a href="#">Coordinate</a>	꼭지점 2

```
polygon.equals(_geometry) => Boolean
```

주어진 Geometry 객체와 같은지 비교한다.

Kind: instance method of [Polygon](#)

Returns: Boolean - true:같음, false:다름

Param	Type	Description
_geometry	<a href="#">Geometry</a>	Geometry 객체

```
polygon.isContains(_geometry) ⇒ Boolean
```

주어진 공간기하객체를 포함하는지 비교한다.

Kind: instance method of [Polygon](#)

Returns: Boolean - 포함하면 true

Param	Type	Description
_geometry	<a href="#">Geometry</a>	Geometry 객체

```
polygon.isWithin(_geometry) ⇒ Boolean
```

주어진 공간기하객체에 포함되는지 비교한다.

Kind: instance method of [Polygon](#)

Returns: Boolean - 포함되면 true

Param	Type	Description
_geometry	<a href="#">Geometry</a>	Geometry 객체

```
polygon.getBoundary() ⇒ Envelope
```

공간기하객체를 포함하는 사각형의 Boundary 영역을 반환한다.

Kind: instance method of [Polygon](#)

Returns: [Envelope](#) - Envelope 영역

`polygon.getCentroid() ⇒ Coordinate`

공간기하객체의 중심좌표를 반환한다.

Kind: instance method of [Polygon](#)

Returns: [Coordinate](#) - 중심좌표

`polygon.minDistance(_coordinate) ⇒ Number`

주어진 좌표와의 최단거리를 반환한다.

Kind: instance method of [Polygon](#)

Returns: Number - 최단거리

Param	Type	Description
<code>_coordinate</code>	<a href="#">Coordinate</a>	좌표

`polygon.distance(_geometry) ⇒ Number`

주어진 공간기하객체와의 중심점 간의 거리를 반환한다.

Kind: instance method of [Polygon](#)

Returns: Number - 거리

Param	Type	Description
_geometry	<a href="#">Geometry</a>	공간 기하 객체

`polygon.getLength() ⇒ Number`

공간기하객체의 길이를 반환한다.

Kind: instance method of [Polygon](#)

Returns: Number - 길이

`polygon.moveCentroid(중심)`

주어진 중심좌표로 공간기하객체를 이동한다.

Kind: instance method of [Polygon](#)

Param	Type	Description
중심	<a href="#">Coordinate</a>	좌표

`polygon.resizeBox(width, height) ⇒ Geometry`

중심좌표는 고정한 채 Bounding Box 의 width, height 를 리사이즈 한다.

Kind: instance method of [Polygon](#)

Returns: [Geometry](#) - 리사이즈된 공간 기하 객체

Param	Type	Description
width	Number	너비
height	Number	높이

```
polygon.fitToBoundary(envelope) ⇒ Geometry
```

주어진 Boundary 영역 안으로 공간 기하 객체를 적용한다.(이동 & 리사이즈)

Kind: instance method of [Polygon](#)

Returns: [Geometry](#) - 적용된 공간 기하 객체

Param	Type	Description
envelope	<a href="#">Envelope</a>	Envelope 영역

```
polygon.convertCoordinate(coordinate) ⇒ Coordinate
```

파라미터가 [x, y] 형식의 좌표 Array 이면 OG.geometry.Coordinate 인스턴스를 new 하여 반환한다.

Kind: instance method of [Polygon](#)

Param	Type	Description
coordinate	<a href="#">Coordinate</a>   Array. <Number>	[x, y] 형식의 좌표 Array 또는 OG.geometry.Coordinate 인스턴스

```
polygon.distanceToLine(p, line) ⇒ Number
```

포인트 P로부터 라인 AB의 거리를 계산한다.

Note: NON-ROBUST!

Kind: instance method of [Polygon](#)

Returns: Number - 거리

Param	Type	Description
-------	------	-------------

p      [Coordinate](#) | Array.<Number> 기준좌표  
line [Array.<Coordinate>](#)                                  라인 시작좌표, 끝좌표 Array

`polygon.distanceLineToLine(line1, line2) ⇒ Number`

라인1로부터 라인2의 거리를 계산한다.

Note: NON-ROBUST!

Kind: instance method of [Polygon](#)

Returns: Number - 거리

Param	Type	Description
line1	<a href="#"><u>Array.&lt;Coordinate&gt;</u></a>	line1 라인 시작좌표, 끝좌표 Array
line2	<a href="#"><u>Array.&lt;Coordinate&gt;</u></a>	line2 라인 시작좌표, 끝좌표 Array

`polygon.intersectToLine(line) ⇒ Array.<Coordinate>`

기하도형이 주어진 라인과 교차하는 좌표들을 반환한다.

Kind: instance method of [Polygon](#)

Param	Type	Description
line	<a href="#"><u>Array.&lt;Coordinate&gt;</u></a>	라인 시작좌표, 끝좌표 Array

`polygon.shortestIntersectToLine(line) ⇒ Array.<Coordinate>`

기하도형이 주어진 라인과 교차하는 좌표중 시작좌표에 가장 가까운 좌표를 반환한다.

Kind: instance method of [Polygon](#)

Param	Type	Description
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 시작좌표, 끝좌표 Array

`polygon.intersectLineToLine(line1, line2, extension) ⇒ Coordinate`

라인1 과 라인2 의 교차점을 계산한다.

Kind: instance method of [Polygon](#)

Returns: [Coordinate](#) - 교차점

Param	Type	Description
line1	<a href="#">Array.&lt;Coordinate&gt;</a>	line1 라인 시작좌표, 끝좌표 Array
line2	<a href="#">Array.&lt;Coordinate&gt;</a>	line2 라인 시작좌표, 끝좌표 Array
extension	boolean	라인을 연장하여 교차점을 계산하는 여부

`polygon.intersectCircleToLine(center, radius, from, to) ⇒ Array.<Coordinate>`

주어진 원과 주어진 라인의 교차점을 계산한다.

Kind: instance method of [Polygon](#)

Returns: [Array.<Coordinate>](#) - 교차점

Param	Type	Description
center	<a href="#">Coordinate</a>	중심점
radius	Number	반경
from	<a href="#">Coordinate</a>	line 라인 시작좌표
to	<a href="#">Coordinate</a>	line 라인 끝좌표

`polygon.intersectPointToLine(p, line) ⇒ Coordinate`

포인트 P로부터 라인 AB 까지 수직인 가상선을 생각할때, 그 교차점을 계산한다.

Note: NON-ROBUST!

Kind: instance method of [Polygon](#)

Returns: [Coordinate](#) - 교차점

Param	Type	Description
p	<a href="#">Coordinate</a>   Array.<Number>	기준좌표
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 시작좌표, 끝좌표 Array

```
polygon.getPercentageDistanceFromPoint(_coordinate) => Object
```

주어진 좌표에 대해 공간기하객체 바운더리 대비 가로, 세로 위치 퍼센테이지 비율을 구한다.

Kind: instance method of [Polygon](#)

Returns: Object - {px , py}

Param	Type	Description
_coordinate	<a href="#">Coordinate</a>	좌표

```
polygon.isContainsPoint(_coordinate) => boolean
```

공간기하객체가 주어진 좌표를 포함하는지를 반환한다.

Kind: instance method of [Polygon](#)

Returns: boolean - true, false

Param	Type	Description
_coordinate	<a href="#">Coordinate</a>	좌표

```
polygon.getPointFromPercentageDistance(pXpY) => Coordinate
```

공간기하객체에 대한 퍼센테이지 좌표의 실제 좌표를 구한다.

Kind: instance method of [Polygon](#)

Returns: [Coordinate](#) - 실 좌표

Param Type Description

pXpY Array 퍼센테이지 좌표

```
polygon.getParallelLine(from, to, distance) => Array.<Coordinate>
```

주어진 선분과 일정 거리에 있는 평행한 선분을 반환한다.

Kind: instance method of [Polygon](#)

Returns: [Array.<Coordinate>](#) - 평행선 시작좌표, 끝좌표 Array

Param Type Description

from [Coordinate](#) 라인 시작좌표

to [Coordinate](#) 라인 끝좌표

distance

```
polygon.getParallelPath(line, distance)
```

주어진 라인과 일정 거리에 있는 평행한 라인을 반환한다.

Kind: instance method of [Polygon](#)

Param Type Description

line [Array.<Coordinate>](#) 라인 좌표 Array

distance

```
polygon.reset()
```

저장된 boundary 를 클리어하여 새로 계산하도록 한다.

Kind: instance method of [Polygon](#)

geometry.Rectangle ← [Polygon](#)

Kind: static class of [geometry](#)

Extends: [Polygon](#)

Requires: module:OG.geometry.Coordinate, module:OG.geometry.Envelope,  
module:OG.geometry.Geometry

Author: [Seungpil Park \(mailto:spspark@uengine.org\)](mailto:Seungpil_Park@uengine.org)

- [.Rectangle](#) ← [Polygon](#)

- [new OG.geometry.Rectangle\(upperLeft, width, height\)](#)
- [.vertices](#) : [Array.<Coordinate>](#)
- [.TYPE](#) : Number
- [.IS\\_CLOSED](#) : Boolean
- [.style](#) : [Style](#)
- [.boundary](#) : [Envelope](#)
- [.getVertices\(\)](#) ⇒ [Array.<Coordinate>](#)
- [.move\(offsetX, offsetY\)](#) ⇒ [Geometry](#)
- [.resize\(upper, lower, left, right\)](#) ⇒ [Geometry](#)
- [.rotate\(angle, origin\)](#) ⇒ [Geometry](#)

- [.toString\(\)](#) ⇒ String
- [.angleBetweenPoints\(prev, next\)](#) ⇒ Number
- [.isRightAngleBetweenPoints\(prev, next\)](#) ⇒ Object
- [.angleBetweenThreePoints\(prev, next\)](#) ⇒ Number
- [.isEqual\(\\_geometry\)](#) ⇒ Boolean
- [.isContains\(\\_geometry\)](#) ⇒ Boolean
- [.isWithin\(\\_geometry\)](#) ⇒ Boolean
- [.getBoundary\(\)](#) ⇒ [Envelope](#)
- [.getCentroid\(\)](#) ⇒ [Coordinate](#)
- [.minDistance\(\\_coordinate\)](#) ⇒ Number
- [.distance\(\\_geometry\)](#) ⇒ Number
- [.getLength\(\)](#) ⇒ Number
- [.moveCentroid\(중심\)](#)
- [.resizeBox\(width, height\)](#) ⇒ [Geometry](#)
- [.fitToBoundary\(envelope\)](#) ⇒ [Geometry](#)
- [.convertCoordinate\(coordinate\)](#) ⇒ [Coordinate](#)
- [.distanceToLine\(p, line\)](#) ⇒ Number
- [.distanceLineToLine\(line1, line2\)](#) ⇒ Number
- [.intersectToLine\(line\)](#) ⇒ [Array.<Coordinate>](#)
- [.shortestIntersectToLine\(line\)](#) ⇒ [Array.<Coordinate>](#)
- [.intersectLineToLine\(line1, line2, extension\)](#) ⇒ [Coordinate](#)
- [.intersectCircleToLine\(center, radius, from, to\)](#) ⇒ [Array.<Coordinate>](#)
- [.intersectPointToLine\(p, line\)](#) ⇒ [Coordinate](#)
- [.getPercentageDistanceFromPoint\(\\_coordinate\)](#) ⇒ Object
- [.isContainsPoint\(\\_coordinate\)](#) ⇒ boolean
- [.getPointFromPercentageDistance\(pXpY\)](#) ⇒ [Coordinate](#)

- [.getParallelLine\(from, to, distance\)](#) ⇒ `Array.<Coordinate>`
- [.getParallelPath\(line, distance\)](#)
- [.reset\(\)](#)

```
new OG.geometry.Rectangle(upperLeft, width, height)
```

`Rectangle` 공간 기하 객체(Spatial Geometry Object)

Param	Type	Description
upperLeft	<a href="#">Coordinate</a>	좌상단좌표
width	Number	너비
height	Number	높이

Example

```
var geom = new OG.geometry.Rectangle([20, 5], 10, 10);
```

```
rectangle.vertices : Array.<Coordinate>
```

Line Vertex 좌표 Array

Kind: instance property of [Rectangle](#)

```
rectangle.TYPE : Number
```

공간 기하 객체 타입

Kind: instance property of [Rectangle](#)

Overrides: [TYPE](#)

rectangle.IS\_CLOSED : Boolean

닫힌 기하 객체 인지 여부

Kind: instance property of [Rectangle](#)

rectangle.style : [Style](#)

스타일 속성

Kind: instance property of [Rectangle](#)

Overrides: [style](#)

rectangle.boundary : [Envelope](#)

공간기하객체를 포함하는 사각형의 Boundary 영역

Kind: instance property of [Rectangle](#)

rectangle.getVertices() => [Array.<Coordinate>](#)

공간기하객체의 모든 꼭지점을 반환한다.

Kind: instance method of [Rectangle](#)

Returns: [Array.<Coordinate>](#) - 꼭지점 좌표 Array

rectangle.move(offsetX, offsetY) ⇒ [Geometry](#)

가로, 세로 Offset 만큼 좌표를 이동한다.

Kind: instance method of [Rectangle](#)

Returns: [Geometry](#) - 이동된 공간 기하 객체

Param	Type	Description
offsetX	Number	가로 Offset
offsetY	Number	세로 Offset

rectangle.resize(upper, lower, left, right) ⇒ [Geometry](#)

상, 하, 좌, 우 외곽선을 이동하여 Envelope 을 리사이즈 한다.

Kind: instance method of [Rectangle](#)

Returns: [Geometry](#) - 리사이즈된 공간 기하 객체

Param	Type	Description
upper	Number	상단 라인 이동 Offset(위 방향으로 +)
lower	Number	하단 라인 이동 Offset(아래 방향으로 +)
left	Number	좌측 라인 이동 Offset(좌측 방향으로 +)
right	Number	우측 라인 이동 Offset(우측 방향으로 +)

rectangle.rotate(angle, origin) ⇒ [Geometry](#)

기준 좌표를 기준으로 주어진 각도 만큼 회전한다.

Kind: instance method of [Rectangle](#)

Returns: [Geometry](#) - 회전된 공간 기하 객체

Param	Type	Description
angle	Number	회전 각도
origin	<a href="#">Coordinate</a>	기준 좌표(default: 중심좌표)

rectangle.toString() ⇒ String

객체 프라퍼티 정보를 JSON 스트링으로 반환한다.

Kind: instance method of [Rectangle](#)

Overrides: [toString](#)

Returns: String - 프라퍼티 정보

rectangle.angleBetweenPoints(prev, next) ⇒ Number

공간기하객체의 두 꼭지점 사이에 가상의 선을 그렸을때, 그 기울기를 구한다.

Kind: instance method of [Rectangle](#)

Returns: Number - 기울기

Param	Type	Description
-------	------	-------------

prev [Coordinate](#) 꼭지점 1

next [Coordinate](#) 꼭지점 2

rectangle.isRightAngleBetweenPoints(prev, next) ⇒ Object

공간기하객체의 두 꼭지점 사이의 기울기가 수평또는 수직인지 판별한다.

Kind: instance method of [Rectangle](#)

Returns: Object - {flag : true or false, type: horizontal or vertical or none}

Param Type Description

prev [Coordinate](#) 꼭지점 1

next [Coordinate](#) 꼭지점 2

```
rectangle.angleBetweenThreePoints(prev, next) => Number
```

공간기하객체의 세 꼭지점 사이의 각도 중 작은 각도를 반환한다.

Kind: instance method of [Rectangle](#)

Returns: Number - 기울기

Param Type Description

prev [Coordinate](#) 꼭지점 1

next [Coordinate](#) 꼭지점 2

```
rectangle.equals(_geometry) => Boolean
```

주어진 [Geometry](#) 객체와 같은지 비교한다.

Kind: instance method of [Rectangle](#)

Returns: Boolean - true:같음, false:다름

Param Type Description

\_geometry [Geometry](#) Geometry 객체

```
rectangle.isContains(_geometry) => Boolean
```

주어진 공간기하 객체를 포함하는지 비교한다.

Kind: instance method of [Rectangle](#)

Returns: Boolean - 포함하면 true

Param	Type	Description
_geometry	<a href="#">Geometry</a>	Geometry 객체

```
rectangle.isWithin(_geometry) => Boolean
```

주어진 공간기하 객체에 포함되는지 비교한다.

Kind: instance method of [Rectangle](#)

Returns: Boolean - 포함되면 true

Param	Type	Description
_geometry	<a href="#">Geometry</a>	Geometry 객체

```
rectangle.getBoundary() => Envelope
```

공간기하 객체를 포함하는 사각형의 Boundary 영역을 반환한다.

Kind: instance method of [Rectangle](#)

Returns: [Envelope](#) - Envelope 영역

```
rectangle.getCentroid() => Coordinate
```

공간기하객체의 중심좌표를 반환한다.

Kind: instance method of [Rectangle](#)

Returns: [Coordinate](#) - 중심좌표

```
rectangle.minDistance(_coordinate) => Number
```

주어진 좌표와의 최단거리를 반환한다.

Kind: instance method of [Rectangle](#)

Returns: Number - 최단거리

Param	Type	Description
_coordinate	<a href="#">Coordinate</a>	좌표

```
rectangle.distance(_geometry) => Number
```

주어진 공간기하객체와의 중심점 간의 거리를 반환한다.

Kind: instance method of [Rectangle](#)

Returns: Number - 거리

Param	Type	Description
_geometry	<a href="#">Geometry</a>	공간 기하 객체

```
rectangle.getLength() => Number
```

공간기하객체의 길이를 반환한다.

Kind: instance method of [Rectangle](#)

Returns: Number - 길이

rectangle.moveToCentroid(중심)

주어진 중심좌표로 공간기하 객체를 이동한다.

Kind: instance method of [Rectangle](#)

Param Type Description

중심 [Coordinate](#) 좌표

rectangle.resizeBox(width, height) => [Geometry](#)

중심좌표는 고정한 채 Bounding Box 의 width, height 를 리사이즈 한다.

Kind: instance method of [Rectangle](#)

Returns: [Geometry](#) - 리사이즈된 공간 기하 객체

Param Type Description

width Number 너비

height Number 높이

rectangle.fitToBoundary(envelope) => [Geometry](#)

주어진 Boundary 영역 안으로 공간 기하 객체를 적용한다. (이동 & 리사이즈)

Kind: instance method of [Rectangle](#)

Returns: [Geometry](#) - 적용된 공간 기하 객체

Param	Type	Description
envelope	<a href="#">Envelope</a>	Envelope 영역

rectangle.convertCoordinate(coordinate) ⇒ [Coordinate](#)

파라미터가 [x, y] 형식의 좌표 Array 이면 OG.geometry.Coordinate 인스턴스를 new 하여 반환한다.

Kind: instance method of [Rectangle](#)

Param	Type	Description
coordinate	<a href="#">Coordinate</a>   Array. <Number>	[x, y] 형식의 좌표 Array 또는 OG.geometry.Coordinate 인스턴스

rectangle.distanceToLine(p, line) ⇒ Number

포인트 P로부터 라인 AB의 거리를 계산한다.

Note: NON-ROBUST!

Kind: instance method of [Rectangle](#)

Returns: Number - 거리

Param	Type	Description
p	<a href="#">Coordinate</a>   Array.<Number>	기준좌표
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 시작좌표, 끝좌표 Array

rectangle.distanceLineToLine(line1, line2) ⇒ Number

라인1로부터 라인2의 거리를 계산한다.

Note: NON-ROBUST!

Kind: instance method of [Rectangle](#)

Returns: Number - 거리

Param Type Description

line1 [Array.<Coordinate>](#) line1 라인 시작좌표, 끝좌표 Array

line2 [Array.<Coordinate>](#) line2 라인 시작좌표, 끝좌표 Array

rectangle.intersectToLine(line) ⇒ [Array.<Coordinate>](#)

기하도형이 주어진 라인과 교차하는 좌표들을 반환한다.

Kind: instance method of [Rectangle](#)

Param Type Description

line [Array.<Coordinate>](#) 라인 시작좌표, 끝좌표 Array

rectangle.shortestIntersectToLine(line) ⇒ [Array.<Coordinate>](#)

기하도형이 주어진 라인과 교차하는 좌표중 시작좌표에 가장 가까운 좌표를 반환한다.

Kind: instance method of [Rectangle](#)

Param Type Description

line [Array.<Coordinate>](#) 라인 시작좌표, 끝좌표 Array

rectangle.intersectLineToLine(line1, line2, extension) ⇒ [Coordinate](#)

라인1 과 라인2 의 교차점을 계산한다.

Kind: instance method of [Rectangle](#)

Returns: [Coordinate](#) - 교차점

Param	Type	Description
line1	<a href="#">Array.&lt;Coordinate&gt;</a>	line1 라인 시작좌표, 끝좌표 Array
line2	<a href="#">Array.&lt;Coordinate&gt;</a>	line2 라인 시작좌표, 끝좌표 Array
extension	boolean	라인을 연장하여 교차점을 계산하는 여부

rectangle.intersectCircleToLine(`center`, `radius`, `from`, `to`) ⇒ [Array.<Coordinate>](#)

주어진 원과 주어진 라인의 교차점을 계산한다.

Kind: instance method of [Rectangle](#)

Returns: [Array.<Coordinate>](#) - 교차점

Param	Type	Description
center	<a href="#">Coordinate</a>	중심점
radius	Number	반경
from	<a href="#">Coordinate</a>	라인 시작좌표
to	<a href="#">Coordinate</a>	라인 끝좌표

rectangle.intersectPointToLine(`p`, `line`) ⇒ [Coordinate](#)

포인트 P로부터 라인 AB 까지 수직인 가상선을 생각할때, 그 교차점을 계산한다.

Note: NON-ROBUST!

Kind: instance method of [Rectangle](#)

Returns: [Coordinate](#) - 교차점

Param	Type	Description
p	<a href="#">Coordinate</a>   <a href="#">Array.&lt;Number&gt;</a>	기준좌표
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 시작좌표, 끝좌표 Array

```
rectangle.getPercentageDistanceFromPoint(_coordinate) ⇒ Object
```

주어진 좌표에 대해 공간기하객체 바운더리 대비 가로, 세로 위치 퍼센테이지 비율을 구한다.

Kind: instance method of [Rectangle](#)

Returns: Object - {px , py}

Param	Type	Description
_coordinate	<a href="#">Coordinate</a>	좌표

```
rectangle.isContainsPoint(_coordinate) ⇒ boolean
```

공간기하객체가 주어진 좌표를 포함하는지를 반환한다.

Kind: instance method of [Rectangle](#)

Returns: boolean - true, false

Param	Type	Description
_coordinate	<a href="#">Coordinate</a>	좌표

```
rectangle.getPointFromPercentageDistance(pXpY) ⇒ Coordinate
```

공간기하객체에 대한 퍼센테이지 좌표의 실제 좌표를 구한다.

Kind: instance method of [Rectangle](#)

Returns: [Coordinate](#) - 실 좌표

Param	Type	Description
-------	------	-------------

## pXpY Array 퍼센테이지 좌표

rectangle.getParallelLine(from, to, distance) ⇒ [Array.<Coordinate>](#)

주어진 선분과 일정 거리에 있는 평행한 선분을 반환한다.

Kind: instance method of [Rectangle](#)

Returns: [Array.<Coordinate>](#) - 평행선 시작좌표, 끝좌표 Array

Param	Type	Description
from	<a href="#">Coordinate</a>	라인 시작좌표
to	<a href="#">Coordinate</a>	라인 끝좌표
distance		

rectangle.getParallelPath(line, distance)

주어진 라인과 일정 거리에 있는 평행한 라인을 반환한다.

Kind: instance method of [Rectangle](#)

Param	Type	Description
line	<a href="#">Array.&lt;Coordinate&gt;</a>	라인 좌표 Array
distance		

rectangle.reset()

저장된 boundary 를 클리어하여 새로 계산하도록 한다.

Kind: instance method of [Rectangle](#)

geometry.Style <= [HashMap](#)

Kind: static class of [geometry](#)

Extends: [HashMap](#)

Author: [Seungpil Park \(mailto:sppark@uengine.org\)](mailto:Seungpil.Park@sppark@uengine.org)

- [.Style](#) <= [HashMap](#)

- [new OG.geometry.Style\(style\)](#)
- [.map](#) : Object
- [.put\(key, value\)](#)
- [.get\(key\)](#) ⇒ Object
- [.containsKey\(key\)](#) ⇒ Boolean
- [.containsValue\(value\)](#) ⇒ Boolean
- [.isEmpty\(\)](#) ⇒ Boolean
- [.clear\(\)](#)
- [.remove\(key\)](#)
- [.keys\(\)](#) ⇒ Array.<String>
- [.values\(\)](#) ⇒ Array.<Object>
- [.size\(\)](#) ⇒ Number
- [.toString\(\)](#) ⇒ String

[new OG.geometry.Style\(style\)](#)

## 스타일(StyleSheet) Property 정보 클래스

Param Type Description  
styleObject 키:값 매핑된 스타일 프라퍼티 정보

### Example

```
var style = new OG.geometry.Style({  
    'cursor': 'default',  
    'stroke': 'black'  
});
```

style.map : Object

key:value 매핑 JSON 오브젝트

Kind: instance property of [Style](#)

style.put(key, value)

key : value 를 매핑한다.

Kind: instance method of [Style](#)

Param Type Description  
key String 키  
value Object 값

style.get(key) ⇒ Object

key 에 대한 value 를 반환한다.

Kind: instance method of [Style](#)

Returns: Object - 값

Param Type Description

key String 키

```
style.containsKey(key) => Boolean
```

주어진 key 를 포함하는지 여부를 반환한다.

Kind: instance method of [Style](#)

Param Type Description

key String 키

```
style.containsValue(value) => Boolean
```

주어진 value 를 포함하는지 여부를 반환한다.

Kind: instance method of [Style](#)

Param Type Description

value Object 값

```
style.isEmpty() => Boolean
```

Empty 여부를 반환한다.

Kind: instance method of [Style](#)

style.clear()

매핑정보를 클리어한다.

Kind: instance method of [Style](#)

style.remove(key)

주어진 key 의 매핑정보를 삭제한다.

Kind: instance method of [Style](#)

Param Type Description

key String 키

style.keys() ⇒ Array.<String>

key 목록을 반환한다.

Kind: instance method of [Style](#)

Returns: Array.<String> - 키목록

style.values() ⇒ Array.<Object>

value 목록을 반환한다.

Kind: instance method of [Style](#)

Returns: Array.<Object> - 값목록

style.size() ⇒ Number

매핑된 key:value 갯수를 반환한다.

Kind: instance method of [Style](#)

style.toString() ⇒ String

객체 프라퍼티 정보를 JSON 스트링으로 반환한다.

Kind: instance method of [Style](#)

Returns: String - 프라퍼티 정보

OG.graph : object

Kind: static namespace of [OG](#)

- [.graph](#) : object

- [.Canvas](#)

- [new OG.graph.Canvas\(container, containerSize, backgroundColor, backgroundImage\)](#)
    - [.initConfig\(config\)](#)
    - [.getRenderer\(\)](#) ⇒ OG.RaphaelRenderer

- [`.getContainer\(\)`](#) ⇒ HTMLElement
- [`.getEventHandler\(\)`](#) ⇒ OG.EventHandler
- [`.addSlider\(\)`](#)
- [`.removeSlider\(\)`](#)
- [`.drawShape\(position, shape, size, style, id, parentId, preventEvent\)`](#) ⇒ Element
- [`.drawTransformer\(position, label, inputs, outputs, id\)`](#) ⇒ Element
- [`.setShapeStyle\(shapeElement, style\)`](#)
- [`.setTextListInController\(shapeElement, textList\)`](#)
- [`.getTextListInController\(shapeElement\)`](#)
- [`.drawLabel\(shapeElement, text, style\)`](#) ⇒ Element
- [`.redrawConnectedEdge\(element\)`](#)
- [`.reconnect\(edge\)`](#) ⇒ Element
- [`.connect\(fromElement, toElement, style, label, fromP, toP, preventTrigger, id, edgeShape\)`](#) ⇒ \* | Element
- [`.connectWithTerminalId\(fromTerminal, toTerminal, style, label\)`](#) ⇒ String | String | geometry
- [`.disconnect\(element\)`](#)
- [`.group\(elements\)`](#) ⇒ Element
- [`.ungroup\(groupElements\)`](#) ⇒ Array.<Element>
- [`.addToGroup\(groupElement, elements\)`](#)
- [`.collapse\(element\)`](#)
- [`.expand\(element\)`](#)
- [`.clear\(\)`](#)
- [`.removeShape\(element\)`](#)
- [`.removeChild\(element\)`](#)
- [`.removeGuide\(element\)`](#)

- [.removeAllGuide\(\)](#)
- [.getRootElement\(\) ⇒ Element](#)
- [.getRootGroup\(\) ⇒ Element](#)
- [.getElementByPoint\(position\) ⇒ Element](#)
- [.getElementsByBBox\(envelope\) ⇒ Array.<Element>](#)
- [.setAttr\(element, attribute\)](#)
- [.getAttr\(element, attrName\) ⇒ Object](#)
- [.toFront\(element\)](#)
- [.toBack\(element\)](#)
- [.bringForward\(element\)](#)
- [.sendBackward\(element\)](#)
- [.getCanvasSize\(\) ⇒ Array.<Number>](#)
- [.setCanvasSize\(size\)](#)
- [.fitCanvasSize\(minSize, fitScale\)](#)
- [.setViewBox\(position, size, isFit\)](#)
- [.getScale\(\) ⇒ Number](#)
- [.setScale\(scale\)](#)
- [.show\(element\)](#)
- [.hide\(element\)](#)
- [.appendChild\(srcElement, targetElement\) ⇒ Element](#)
- [.insertAfter\(srcElement, targetElement\) ⇒ Element](#)
- [.insertBefore\(srcElement, targetElement\) ⇒ Element](#)
- [.move\(element, offset\) ⇒ Element](#)
- [.moveCentroid\(element, position\) ⇒ Element](#)
- [.rotate\(element, angle\) ⇒ Element](#)
- [.resize\(element, offset\) ⇒ Element](#)

- [`.resizeBox\(element, size\)`](#) ⇒ Element
- [`.clone\(element\)`](#) ⇒ Element
- [`.getBoundary\(element\)`](#) ⇒ Envelope
- [`.getElementById\(id\)`](#) ⇒ Element
- [`.getElementsByType\(shapeType, excludeType\)`](#) ⇒ Array.  
`<Element>`
- [`.getElementsByShapeId\(shapeId\)`](#) ⇒ Array.`<Element>`
- [`.getRelatedElementsFromEdge\(edgeElement\)`](#) ⇒ Object
- [`.getParent\(Element\)`](#) ⇒ Element
- [`.getchilds\(element\)`](#) ⇒ Array
- [`.getAllShapes\(\)`](#) ⇒ Array
- [`.getAllEdges\(\)`](#) ⇒ Array
- [`.getBBox\(element\)`](#) ⇒ Object
- [`.getRootBBox\(\)`](#) ⇒ Object
- [`.getRealRootBBox\(\)`](#) ⇒ Object
- [`.isSVG\(\)`](#) ⇒ Boolean
- [`.isVML\(\)`](#) ⇒ Boolean
- [`.setCustomData\(shapeElement, data\)`](#)
- [`.getCustomData\(shapeElement\)`](#) ⇒ Object
- [`.setExtCustomData\(shapeElement, data\)`](#)
- [`.getExtCustomData\(shapeElement\)`](#) ⇒ Object
- [`.toXML\(\)`](#) ⇒ String
- [`.toJSON\(\)`](#) ⇒ Object
- [`.loadXML\(xml\)`](#) ⇒ Object
- [`.loadJSON\(json\)`](#) ⇒ Object
- [`.undo\(\)`](#)

- [.redo\(\)](#)
- [.getPrevEdges\(element\)](#) ⇒ Array.<Element>
- [.getNextEdges\(element\)](#) ⇒ Array.<Element>
- [.getPrevShapes\(element\)](#) ⇒ Array.<Element>
- [.getPrevShapeIds\(element\)](#) ⇒ Array.<String>
- [.getNextShapes\(element\)](#) ⇒ Array.<Element>
- [.getNextShapeIds\(element\)](#) ⇒ Array.<String>
- [.onDrawShape\(callbackFunc\)](#)
- [.onUndo\(callbackFunc\)](#)
- [.onRedo\(callbackFunc\)](#)
- [.onDivideLane\(callbackFunc\)](#)
- [.onDrawLabel\(callbackFunc\)](#)
- [.onLabelChanged\(callbackFunc\)](#)
- [.onBeforeLabelChange\(callbackFunc\)](#)
- [.onRedrawShape\(callbackFunc\)](#)
- [.onRemoveShape\(callbackFunc\)](#)
- [.onRotateShape\(callbackFunc\)](#)
- [.onMoveShape\(callbackFunc\)](#)
- [.onResizeShape\(callbackFunc\)](#)
- [.onBeforeConnectShape\(callbackFunc\)](#)
- [.onBeforeRemoveShape\(callbackFunc\)](#)
- [.onConnectShape\(callbackFunc\)](#)
- [.onDisconnectShape\(callbackFunc\)](#)
- [.onGroup\(callbackFunc\)](#)
- [.onUnGroup\(callbackFunc\)](#)
- [.onCollapsed\(callbackFunc\)](#)

- [.onExpanded\(callbackFunc\)](#)

graph.Canvas

Kind: static class of [graph](#)

Requires: module:OG.common.\* , module:OG.geometry.\* , module:OG.shape.\* , module:OG.renderer.\* , module:OG.handler.\* , module:OG.layout.\* , module:raphael-2.1.0

Author: [Seungpil Park \(mailto:spspark@uengine.org\)](mailto:Seungpil.Park@uengine.org)

- [.Canvas](#)

- [new OG.graph.Canvas\(container, containerSize, backgroundColor, backgroundImage\)](#)
- [.initConfig\(config\)](#)
- [.getRenderer\(\)](#) ⇒ OG.RaphaelRenderer
- [.getContainer\(\)](#) ⇒ HTMLElement
- [.getEventHandler\(\)](#) ⇒ OG.EventHandler
- [.addSlider\(\)](#)
- [.removeSlider\(\)](#)
- [.drawShape\(position, shape, size, style, id, parentId, preventEvent\)](#) ⇒ Element
- [.drawTransformer\(position, label, inputs, outputs, id\)](#) ⇒ Element
- [.setShapeStyle\(shapeElement, style\)](#)
- [.setTextListInController\(shapeElement, textList\)](#)
- [.getTextListInController\(shapeElement\)](#)

- [.drawLabel\(shapeElement, text, style\)](#) ⇒ Element
- [.redrawConnectedEdge\(element\)](#)
- [.reconnect\(edge\)](#) ⇒ Element
- [.connect\(fromElement, toElement, style, label, fromP, toP, preventTrigger, id, edgeShape\)](#) ⇒ \* | Element
- [.connectWithTerminalId\(fromTerminal, toTerminal, style, label\)](#) ⇒ String | String | [geometry](#)
- [.disconnect\(element\)](#)
- [.group\(elements\)](#) ⇒ Element
- [.ungroup\(groupElements\)](#) ⇒ Array.<Element>
- [.addToGroup\(groupElement, elements\)](#)
- [.collapse\(element\)](#)
- [.expand\(element\)](#)
- [.clear\(\)](#)
- [.removeShape\(element\)](#)
- [.removeChild\(element\)](#)
- [.removeGuide\(element\)](#)
- [.removeAllGuide\(\)](#)
- [.getRootElement\(\)](#) ⇒ Element
- [.getRootGroup\(\)](#) ⇒ Element
- [.getElementByPoint\(position\)](#) ⇒ Element
- [.getElementsByBBox\(envelope\)](#) ⇒ Array.<Element>
- [.setAttr\(element, attribute\)](#)
- [.getAttr\(element, attrName\)](#) ⇒ Object
- [.toFront\(element\)](#)
- [.toBack\(element\)](#)
- [.bringForward\(element\)](#)

- [.sendBackward\(element\)](#)
- [.getCanvasSize\(\)](#) ⇒ Array.<Number>
- [.setCanvasSize\(size\)](#)
- [.fitCanvasSize\(minSize, fitScale\)](#)
- [.setViewBox\(position, size, isFit\)](#)
- [.getScale\(\)](#) ⇒ Number
- [.setScale\(scale\)](#)
- [.show\(element\)](#)
- [.hide\(element\)](#)
- [.appendChild\(srcElement, targetElement\)](#) ⇒ Element
- [.insertAfter\(srcElement, targetElement\)](#) ⇒ Element
- [.insertBefore\(srcElement, targetElement\)](#) ⇒ Element
- [.move\(element, offset\)](#) ⇒ Element
- [.moveCentroid\(element, position\)](#) ⇒ Element
- [.rotate\(element, angle\)](#) ⇒ Element
- [.resize\(element, offset\)](#) ⇒ Element
- [.resizeBox\(element, size\)](#) ⇒ Element
- [.clone\(element\)](#) ⇒ Element
- [.getBoundary\(element\)](#) ⇒ Envelope
- [.getElementById\(id\)](#) ⇒ Element
- [.getElementsByType\(shapeType, excludeType\)](#) ⇒ Array.<Element>
- [.getElementsByShapeId\(shapeId\)](#) ⇒ Array.<Element>
- [.getRelatedElementsFromEdge\(edgeElement\)](#) ⇒ Object
- [.getParent\(Element\)](#) ⇒ Element
- [.getchilds\(element\)](#) ⇒ Array
- [.getAllShapes\(\)](#) ⇒ Array

- [.getAllEdges\(\)](#) ⇒ Array
- [.getBBox\(element\)](#) ⇒ Object
- [.getRootBBox\(\)](#) ⇒ Object
- [.getRealRootBBox\(\)](#) ⇒ Object
- [.isSVG\(\)](#) ⇒ Boolean
- [.isVML\(\)](#) ⇒ Boolean
- [.setCustomData\(shapeElement, data\)](#)
- [.getCustomData\(shapeElement\)](#) ⇒ Object
- [.setExtCustomData\(shapeElement, data\)](#)
- [.getExtCustomData\(shapeElement\)](#) ⇒ Object
- [.toXML\(\)](#) ⇒ String
- [.toJSON\(\)](#) ⇒ Object
- [.loadXML\(xml\)](#) ⇒ Object
- [.loadJSON\(json\)](#) ⇒ Object
- [.undo\(\)](#)
- [.redo\(\)](#)
- [.getPrevEdges\(element\)](#) ⇒ Array.<Element>
- [.getNextEdges\(element\)](#) ⇒ Array.<Element>
- [.getPrevShapes\(element\)](#) ⇒ Array.<Element>
- [.getPrevShapeIds\(element\)](#) ⇒ Array.<String>
- [.getNextShapes\(element\)](#) ⇒ Array.<Element>
- [.getNextShapeIds\(element\)](#) ⇒ Array.<String>
- [.onDrawShape\(callbackFunc\)](#)
- [.onUndo\(callbackFunc\)](#)
- [.onRedo\(callbackFunc\)](#)
- [.onDivideLane\(callbackFunc\)](#)

- [.onDrawLabel\(callbackFunc\)](#)
- [.onLabelChanged\(callbackFunc\)](#)
- [.onBeforeLabelChange\(callbackFunc\)](#)
- [.onRedrawShape\(callbackFunc\)](#)
- [.onRemoveShape\(callbackFunc\)](#)
- [.onRotateShape\(callbackFunc\)](#)
- [.onMoveShape\(callbackFunc\)](#)
- [.onResizeShape\(callbackFunc\)](#)
- [.onBeforeConnectShape\(callbackFunc\)](#)
- [.onBeforeRemoveShape\(callbackFunc\)](#)
- [.onConnectShape\(callbackFunc\)](#)
- [.onDisconnectShape\(callbackFunc\)](#)
- [.onGroup\(callbackFunc\)](#)
- [.onUnGroup\(callbackFunc\)](#)
- [.onCollapsed\(callbackFunc\)](#)
- [.onExpanded\(callbackFunc\)](#)

```
new OG.graph.Canvas(container, containerSize, backgroundColor, backgroundImage)
```

OpenGraph 캔버스 클래스

Param	Type	Description
container	HTMLElement   String	컨테이너 DOM element or ID
containerSize	Array.<Number>	컨테이너 Width, Height
backgroundColor	String	캔버스 배경색
backgroundImage	String	캔버스 배경이미지

## Example

```
var canvas = new OG.Canvas('canvas', [1000, 800], 'white',
'url("./images/grid.gif"));

var circleShape = canvas.drawShape([100, 100], new OG.CircleShape(), [100,
100]);
var ellipseShape = canvas.drawShape([300, 200], new OG.EllipseShape('label'),
[100, 50]);

var edge = canvas.connect(circleShape, ellipseShape);
```

```
canvas.initConfig(config)
```

Canvas 의 설정값을 초기화한다.

- selectable	: 클릭선택 가능여부(디폴트 true)
- dragSelectable	: 마우스드래그선택 가능여부(디폴트 true)
- movable	: 이동 가능여부(디폴트 true)
- resizable	: 리사이즈 가능여부(디폴트 true)
- connectable	: 연결 가능여부(디폴트 true)
- selfConnectable	: Self 연결 가능여부(디폴트 true)
- connectCloneable	: 드래그하여 연결시 대상 없을 경우 자동으로 Shape 복사하여 연결 처리 여부 (디폴트 true)
- connectRequired	: 드래그하여 연결시 연결대상 있는 경우에만 Edge 드로잉 처리 여부(디폴트 true)
- labelEditable	: 라벨 수정여부(디폴트 true)
- groupDropable	: 그룹핑 가능여부(디폴트 true)
- enableHotKey	: 헫키 가능여부(디폴트 true)
- enableContextMenu	: 마우스 우클릭 메뉴 가능여부(디폴트 true)
- autoExtensional	: 캔버스 자동 확장 기능(디폴트 true)
- useSlider	: 확대축소 슬라이더 사용 여부
- stickGuide	: 스틱 가이드 표시 여부
- checkBridgeEdge	: 연결된 두 오브젝트의 소속에 따른 연결선 스타일 변화 여부

Kind: instance method of [Canvas](#)

Param	Type	Description
config	Object	JSON 포맷의 configuration

```
canvas.getRenderer() => OG.RaphaelRenderer
```

랜더러를 반환한다.

Kind: instance method of [Canvas](#)

```
canvas.getContainer() => HTMLElement
```

컨테이너 DOM element 를 반환한다.

Kind: instance method of [Canvas](#)

```
canvas.getEventHandler() => OG.EventHandler
```

이벤트 핸들러를 반환한다.

Kind: instance method of [Canvas](#)

```
canvas.addSlider()
```

확대 축소 슬라이더를 설치한다.

Kind: instance method of [Canvas](#)

```
canvas.removeSlider()
```

확대 축소 슬라이더를 삭제한다.

Kind: instance method of [Canvas](#)

```
canvas.drawShape(position, shape, size, style, id, parentId, preventEvent) => Element
```

Shape 을 캔버스에 위치 및 사이즈 지정하여 드로잉한다.

Kind: instance method of [Canvas](#)

Returns: Element - Group DOM Element with geometry

Param	Type	Description
position	Array.<Number>	드로잉할 위치 좌표(중앙 기준)
shape	<a href="#">IShape</a>	Shape
size	Array.<Number>	Shape Width, Height
style	<a href="#">Style</a>   Object	스타일 (Optional)
id	String	Element ID 지정 (Optional)
parentId	String	부모 Element ID 지정 (Optional)
preventEvent	Boolean	이벤트 생성 방지

Example

```
canvas.drawShape([100, 100], new OG.CircleShape(), [50, 50], {stroke: 'red'});
```

```
canvas.drawTransformer(position, label, inputs, outputs, id) => Element
```

Transformer Shape 을 캔버스에 위치 및 사이즈 지정하여 드로잉한다.

Kind: instance method of [Canvas](#)

Returns: Element - Group DOM Element with geometry

Param	Type	Description
positionArray	<Number>	드로잉할 위치 좌표(중앙 기준)
label	String	Label
inputs	Array.<String>	인풋에 위치할 리스트
outputs	Array.<String>	아웃풋에 위치할 리스트
id	String	Element ID 지정 (Optional)

### Example

```
canvas.drawTransformer([100, 100], 'label' ['str1','str2'], ['out']);
```

```
canvas.setShapeStyle(shapeElement, style)
```

Shape 의 스타일을 변경한다.

Kind: instance method of [Canvas](#)

Param	Type	Description
shapeElement	Element Shape DOM element	
style	Object	스타일

```
canvas.setTextListInController(shapeElement, textList)
```

Shape 의 선 연결 커스텀 컨트롤러를 설정한다.

Kind: instance method of [Canvas](#)

Param	Type	Description
shapeElement	Element Shape DOM element	
textList	Array	텍스트 리스트

```
canvas.getTextListInController(shapeElement)
```

Shape 의 선 연결 커스텀 컨트롤러를 가져온다.

Kind: instance method of [Canvas](#)

Param	Type	Description
shapeElement	Element Shape	DOM element

```
canvas.drawLabel(shapeElement, text, style) => Element
```

Shape 의 Label 을 캔버스에 위치 및 사이즈 지정하여 드로잉한다.

Kind: instance method of [Canvas](#)

Returns: Element - DOM Element

Param	Type	Description
shapeElement	Element   String	Shape DOM element or ID
text	String	텍스트
style	<a href="#">Style</a>   Object	스타일

```
canvas.redrawConnectedEdge(element)
```

Shape 의 연결된 Edge 를 redraw 한다.(이동 또는 리사이즈)

Kind: instance method of [Canvas](#)

Param	Type
element	Element

```
canvas.reconnect(edge) => Element
```

연결된 터미널의 vertices 를 초기화한다.

Kind: instance method of [Canvas](#)

Returns: Element - 연결된 Edge 엘리먼트

Param	Type	Description
edge	Element	Edge Shape

```
canvas.connect(fromElement, toElement, style, label, fromP, toP, preventTrigger, id, edgeShape) => *  
| Element
```

두개의 Shape 을 Edge 로 연결한다.

Kind: instance method of [Canvas](#)

Param	Type	Description
fromElement	Element	from Shape Element
toElement	Element	to Shape Element
style	<a href="#">Style</a>   Object	스타일
label	String	Label
fromP		fromElement 와 연결될 터미널 좌표 <a href="#">x,y (optional)</a>
toP		toElement 와 연결될 터미널 좌표 <a href="#">x,y (optional)</a>
preventTrigger		참 일 경우 이벤트 발생을 방지
id		연결선의 아이디
edgeShape	Element	이 값이 없으면 신규 OG.EdgeShape 를 생성

```
canvas.connectWithTerminalId(fromTerminal, toTerminal, style, label) => String | String | geometry
```

두개의 터미널 아이디로 부터 얻어진 Shape를 Edge 로 연결한다.

Kind: instance method of [Canvas](#)

Returns: String - id 부여 할 아이디String - shapeId shapeId[geometry](#) - geom Edge

geometry

Param	Type	Description
fromTerminal	String	from Terminal Id
toTerminal	String	to Terminal Id
style	<a href="#">Style</a>   Object	스타일
label	String	Label

canvas.disconnect(element)

연결속성정보를 삭제한다. Edge 인 경우는 라인만 삭제하고, 일반 Shape 인 경우는 연결된 모든 Edge 를 삭제한다.

Kind: instance method of [Canvas](#)

Param	Type
element	Element

canvas.group(elements) ⇒ Element

주어진 Shape 들을 그룹핑한다.

Kind: instance method of [Canvas](#)

Returns: Element - Group Shape Element

Param	Type
elements	Array.<Element>

canvas.ungroup(groupElements) ⇒ Array.<Element>

주어진 그룹들을 그룹해제한다.

Kind: instance method of [Canvas](#)

Returns: Array.<Element> - ungrouped Elements

Param	Type
groupElements	Array.<Element>

```
canvas.addToGroup(groupElement, elements)
```

주어진 Shape 들을 그룹에 추가한다.

Kind: instance method of [Canvas](#)

Param	Type
groupElement	Element
elements	Array.<Element>

```
canvas.collapse(element)
```

주어진 Shape 이 그룹인 경우 collapse 한다.

Kind: instance method of [Canvas](#)

Param	Type
element	Element

```
canvas.expand(element)
```

주어진 Shape 이 그룹인 경우 expand 한다.

Kind: instance method of [Canvas](#)

Param Type  
element Element

```
canvas.clear()
```

드로잉된 모든 오브젝트를 클리어한다.

Kind: instance method of [Canvas](#)

```
canvas.removeShape(element)
```

Shape 을 캔버스에서 관련된 모두를 삭제한다.

Kind: instance method of [Canvas](#)

Param Type Description  
element Element | String Element 또는 ID

```
canvas.removeChild(element)
```

하위 엘리먼트만 제거한다.

Kind: instance method of [Canvas](#)

Param Type Description  
element Element | String Element 또는 ID

```
canvas.removeGuide(element)
```

ID에 해당하는 Element 의 Move & Resize 용 가이드를 제거한다.

Kind: instance method of [Canvas](#)

Param	Type	Description
element	Element   String	Element 또는 ID

```
canvas.removeAllGuide()
```

모든 Move & Resize 용 가이드를 제거한다.

Kind: instance method of [Canvas](#)

```
canvas.getRootElement() => Element
```

랜더러 캔버스 Root Element 를 반환한다.

Kind: instance method of [Canvas](#)

Returns: Element - Element

```
canvas.getRootGroup() => Element
```

랜더러 캔버스 Root Group Element 를 반환한다.

Kind: instance method of [Canvas](#)

Returns: Element - Element

```
canvas.getElementByPoint(position) => Element
```

주어진 지점을 포함하는 Top Element 를 반환한다.

Kind: instance method of [Canvas](#)

Returns: Element - Element

Param	Type	Description
positionArray.<Number>	위치 좌표	

```
canvas.getElementsByBBox(envelope) => Array.<Element>
```

주어진 Boundary Box 영역에 포함되는 Shape(GEOM, TEXT, IMAGE) Element 를 반환한다.

모든 vertices를 포함한 엘리먼트를 반환한다.

Kind: instance method of [Canvas](#)

Returns: Array.<Element> - Element

Param	Type	Description
envelope	<a href="#">Envelope</a>	Boundary Box 영역

```
canvas.setAttr(element, attribute)
```

엘리먼트에 속성값을 설정한다.

Kind: instance method of [Canvas](#)

Param	Type	Description
element	Element   String	Element 또는 ID

`attributeObject` 속성값

`canvas.getAttr(element, attrName) ⇒ Object`

엘리먼트 속성값을 반환한다.

Kind: instance method of [Canvas](#)

Returns: Object - attribute 속성값

Param	Type	Description
<code>element</code>	<code>Element   String</code>	<code>Element</code> 또는 ID
<code>attrName</code>	<code>String</code>	속성이름

`canvas.toFront(element)`

ID에 해당하는 Element 를 최상단 레이어로 이동한다.

Kind: instance method of [Canvas](#)

Param	Type	Description
<code>element</code>	<code>Element   String</code>	<code>Element</code> 또는 ID

`canvas.toBack(element)`

ID에 해당하는 Element 를 최하단 레이어로 이동한다.

Kind: instance method of [Canvas](#)

Param	Type	Description
<code>element</code>	<code>Element   String</code>	<code>Element</code> 또는 ID

```
canvas.bringForward(element)
```

ID에 해당하는 Element 를 앞으로 한단계 이동한다.

Kind: instance method of [Canvas](#)

Param	Type	Description
element	Element   String	Element 또는 ID

```
canvas.sendBackward(element)
```

ID에 해당하는 Element 를 뒤로 한단계 이동한다.

Kind: instance method of [Canvas](#)

Param	Type	Description
element	Element   String	Element 또는 ID

```
canvas.getCanvasSize() => Array.<Number>
```

랜더러 캔버스의 사이즈(Width, Height)를 반환한다.

Kind: instance method of [Canvas](#)

Returns: Array.<Number> - Canvas Width, Height

```
canvas.setCanvasSize(size)
```

랜더러 캔버스의 사이즈(Width, Height)를 변경한다.

Kind: instance method of [Canvas](#)

Param	Type	Description
size	Array.<Number>	Canvas Width, Height

```
canvas.fitCanvasSize(minSize, fitScale)
```

랜더러 캔버스의 사이즈(Width, Height)를 실제 존재하는 Shape 의 영역에 맞게 변경한다.

Kind: instance method of [Canvas](#)

Param	Type	Description
minSize	Array.<Number>	Canvas 최소 Width, Height
fitScale	Boolean	주어진 minSize 에 맞게 fit 여부(Default:false)

```
canvas.setViewBox(position, size, isFit)
```

새로운 View Box 영역을 설정한다. (ZoomIn & ZoomOut 가능)

Kind: instance method of [Canvas](#)

Param	Type	Description
position	Array.<Number>	위치 좌표(좌상단 기준)
size	Array.<Number>	Canvas Width, Height
isFit	Boolean	Fit 여부

```
canvas.getScale() => Number
```

Scale 을 반환한다. (리얼 사이즈 : Scale = 1)

Kind: instance method of [Canvas](#)

Returns: Number - 스케일값

canvas.setScale(scale)

Scale 을 설정한다. (리얼 사이즈 : Scale = 1)

Kind: instance method of [Canvas](#)

Param Type Description

scale Number 스케일값

canvas.show(element)

ID에 해당하는 Element 를 캔버스에서 show 한다.

Kind: instance method of [Canvas](#)

Param Type Description  
element Element | String Element 또는 ID

canvas.hide(element)

ID에 해당하는 Element 를 캔버스에서 hide 한다.

Kind: instance method of [Canvas](#)

Param Type Description  
element Element | String Element 또는 ID

```
canvas.appendChild(srcElement, targetElement) => Element
```

Source Element 를 Target Element 아래에 append 한다.

Kind: instance method of [Canvas](#)

Returns: Element - Source Element

Param	Type	Description
srcElement	Element   String	Element 또는 ID
targetElement	Element   String	Element 또는 ID

```
canvas.insertAfter(srcElement, targetElement) => Element
```

Source Element 를 Target Element 이후에 insert 한다.

Kind: instance method of [Canvas](#)

Returns: Element - Source Element

Param	Type	Description
srcElement	Element   String	Element 또는 ID
targetElement	Element   String	Element 또는 ID

```
canvas.insertBefore(srcElement, targetElement) => Element
```

Source Element 를 Target Element 이전에 insert 한다.

Kind: instance method of [Canvas](#)

Returns: Element - Source Element

Param	Type	Description
srcElement	Element   String	Element 또는 ID
targetElement	Element   String	Element 또는 ID

```
canvas.move(element, offset) => Element
```

해당 Element 를 가로, 세로 Offset 만큼 이동한다.

Kind: instance method of [Canvas](#)

Returns: Element - Element

Param	Type	Description
element	Element   String	Element 또는 ID
offset	Array.<Number>	[가로, 세로]

```
canvas.moveCentroid(element, position) => Element
```

주어진 중심좌표로 해당 Element 를 이동한다.

Kind: instance method of [Canvas](#)

Returns: Element - Element

Param	Type	Description
element	Element   String	Element 또는 ID
position	Array.<Number>	[x, y]

```
canvas.rotate(element, angle) => Element
```

중심 좌표를 기준으로 주어진 각도 만큼 회전한다.

Kind: instance method of [Canvas](#)

Returns: Element - Element

Param	Type	Description
element	Element   String	Element 또는 ID
angle	Number	각도

```
canvas.resize(element, offset) => Element
```

상, 하, 좌, 우 외곽선을 이동한 만큼 리사이즈 한다.

Kind: instance method of [Canvas](#)

Returns: Element - Element

Param	Type	Description
element	Element   String	Element 또는 ID
offset	Array.<Number>	[상, 하, 좌, 우] 각 방향으로 + 값

```
canvas.resizeBox(element, size) => Element
```

중심좌표는 고정한 채 Bounding Box 의 width, height 를 리사이즈 한다.

Kind: instance method of [Canvas](#)

Returns: Element - Element

Param	Type	Description
element	Element   String	Element 또는 ID
size	Array.<Number>	[Width, Height]

```
canvas.clone(element) => Element
```

노드 Element 를 복사한다.

Kind: instance method of [Canvas](#)

Returns: Element - Element

Param	Type	Description
element	Element   String	Element 또는 ID

`canvas.getBoundary(element) ⇒ Envelope`

ID에 해당하는 Element 의 바운더리 영역을 리턴한다.

Kind: instance method of [Canvas](#)

Returns: [Envelope](#) - Envelope 영역

Param	Type	Description
element	Element   String	Element 또는 ID

`canvas.getElementById(id) ⇒ Element`

ID로 Node Element 를 반환한다.

Kind: instance method of [Canvas](#)

Returns: Element - Element

Param	Type
id	String

`canvas.getElementsByType(shapeType, excludeType) ⇒ Array.<Element>`

Shape 타입에 해당하는 Node Element 들을 반환한다.

Kind: instance method of [Canvas](#)

Returns: Array.<Element> - Element's Array

Param	Type	Description
shapeType	String	Shape 타입(GEOM, HTML, IMAGE, EDGE, GROUP), Null 이면 모든 타입
excludeType	String	제외 할 타입

```
canvas.getElementsByShapeId(shapeId) => Array.<Element>
```

Shape ID에 해당하는 Node Element 들을 반환한다.

Kind: instance method of [Canvas](#)

Returns: Array.<Element> - Element's Array

Param	Type	Description
shapeId	String	Shape ID

```
canvas.getRelatedElementsFromEdge(edgeElement) => Object
```

Edge 엘리먼트와 연결된 fromShape, toShape 엘리먼트를 반환한다.

Kind: instance method of [Canvas](#)

Param	Type	Description
edgeElement	Element   String	Element 또는 ID

```
canvas.getParent(Element) => Element
```

부모 엘리먼트를 반환한다. 부모가 루트일때는 반환하지 않는다.

Kind: instance method of [Canvas](#)

Returns: Element - Element 엘리먼트

Param Type Description  
Element Element 엘리먼트

canvas.getChilds(element) ⇒ Array

그룹의 하위 엘리먼트를 반환한다.

Kind: instance method of [Canvas](#)

Returns: Array - Elements

Param Type Description  
element Element 엘리먼트

canvas.getAllShapes() ⇒ Array

캔버스의 모든 Shape 들을 리턴

Kind: instance method of [Canvas](#)

Returns: Array - Elements

canvas.getAllEdges() ⇒ Array

캔버스의 모든 Edge를 리턴

Kind: instance method of [Canvas](#)

Returns: Array - Edge Elements

```
canvas.getBBox(element) => Object
```

해당 엘리먼트의 BoundingBox 영역 정보를 반환한다.

Kind: instance method of [Canvas](#)

Returns: Object - {width, height, x, y, x2, y2}

Param	Type
element	Element   String

```
canvas.getRootBBox() => Object
```

부모노드기준으로 캔버스 루트 엘리먼트의 BoundingBox 영역 정보를 반환한다.

Kind: instance method of [Canvas](#)

Returns: Object - {width, height, x, y, x2, y2}

```
canvas.getRealRootBBox() => Object
```

부모노드기준으로 캔버스 루트 엘리먼트의 실제 Shape 이 차지하는 BoundingBox 영역 정보를 반환한다.

Kind: instance method of [Canvas](#)

Returns: Object - {width, height, x, y, x2, y2}

```
canvas.isSVG() => Boolean
```

SVG 인지 여부를 반환한다.

Kind: instance method of [Canvas](#)

Returns: Boolean - svg 여부

```
canvas.isVML() => Boolean
```

VML 인지 여부를 반환한다.

Kind: instance method of [Canvas](#)

Returns: Boolean - vml 여부

```
canvas.setCustomData(shapeElement, data)
```

주어진 Shape 엘리먼트에 커스텀 데이터를 저장한다.

Kind: instance method of [Canvas](#)

Param	Type	Description
shapeElement	Element   String	Shape DOM Element or ID
data	Object	JSON 포맷의 Object

```
canvas.getCustomData(shapeElement) => Object
```

주어진 Shape 엘리먼트에 저장된 커스텀 데이터를 반환한다.

Kind: instance method of [Canvas](#)

Returns: Object - JSON 포맷의 Object

Param	Type	Description
shapeElement	Element   String Shape	DOM Element or ID

```
canvas.setExtCustomData(shapeElement, data)
```

주어진 Shape 엘리먼트에 확장 커스텀 데이터를 저장한다.

Kind: instance method of [Canvas](#)

Param	Type	Description
shapeElement	Element   String Shape	DOM Element or ID
data	Object	JSON 포맷의 Object

```
canvas.getExtCustomData(shapeElement) => Object
```

주어진 Shape 엘리먼트에 저장된 확장 커스텀 데이터를 반환한다.

Kind: instance method of [Canvas](#)

Returns: Object - JSON 포맷의 Object

Param	Type	Description
shapeElement	Element   String Shape	DOM Element or ID

```
canvas.toXML() => String
```

Canvas 에 그려진 Shape 들을 OpenGraph XML 문자열로 export 한다.

Kind: instance method of [Canvas](#)

Returns: String - XML 문자열

canvas.toJSONString() ⇒ Object

Canvas에 그려진 Shape들을 OpenGraph JSON 객체로 export 한다.

Kind: instance method of [Canvas](#)

Returns: Object - JSON 포맷의 Object

canvas.loadXML(xml) ⇒ Object

OpenGraph XML 문자열로 부터 Shape를 드로잉한다.

Kind: instance method of [Canvas](#)

Returns: Object - {width, height, x, y, x2, y2}

Param	Type	Description
xml	String   Element	XML 문자열 또는 DOM Element

canvas.loadJSON(json) ⇒ Object

JSON 객체로 부터 Shape를 드로잉한다.

Kind: instance method of [Canvas](#)

Returns: Object - {width, height, x, y, x2, y2}

Param	Type	Description
json	Object	JSON 포맷의 Object

canvas.undo()

캔버스 undo.

Kind: instance method of [Canvas](#)

canvas.redo()

캔버스 redo.

Kind: instance method of [Canvas](#)

canvas.getPrevEdges(element) ⇒ Array.<Element>

연결된 이전 Edge Element 들을 반환한다.

Kind: instance method of [Canvas](#)

Returns: Array.<Element> - Previous Element's Array

Param	Type	Description
element	Element   String	Element 또는 ID

canvas.getNextEdges(element) ⇒ Array.<Element>

연결된 이후 Edge Element 들을 반환한다.

Kind: instance method of [Canvas](#)

Returns: Array.<Element> - Previous Element's Array

Param	Type	Description
element	Element   String	Element 또는 ID

```
canvas.getPrevShapes(element) => Array.<Element>
```

연결된 이전 노드 Element 들을 반환한다.

Kind: instance method of [Canvas](#)

Returns: Array.<Element> - Previous Element's Array

Param	Type	Description
element	Element   String	Element 또는 ID

```
canvas.getPrevShapeIds(element) => Array.<String>
```

연결된 이전 노드 Element ID들을 반환한다.

Kind: instance method of [Canvas](#)

Returns: Array.<String> - Previous Element Id's Array

Param	Type	Description
element	Element   String	Element 또는 ID

```
canvas.getNextShapes(element) => Array.<Element>
```

연결된 이후 노드 Element 들을 반환한다.

Kind: instance method of [Canvas](#)

Returns: Array.<Element> - Previous Element's Array

Param	Type	Description
element	Element   String	Element 또는 ID

```
canvas.getNextShapeIds(element) => Array.<String>
```

연결된 이후 노드 Element ID들을 반환한다.

Kind: instance method of [Canvas](#)

Returns: Array.<String> - Previous Element Id's Array

Param	Type	Description
element	Element   String	Element 또는 ID

```
canvas.onDrawShape(callbackFunc)
```

Shape 이 처음 Draw 되었을 때의 이벤트 리스너

Kind: instance method of [Canvas](#)

Param	Type	Description
callbackFunc	function	콜백함수(event, shapeElement)

```
canvas.onUndo(callbackFunc)
```

Undo 되었을때의 이벤트 리스너

Kind: instance method of [Canvas](#)

Param	Type	Description
callbackFunc	function	콜백함수(event)

```
canvas.onRedo(callbackFunc)
```

Redo 되었을때의 이벤트 리스너

Kind: instance method of [Canvas](#)

Param	Type	Description
callbackFunc	function	콜백함수(event)

```
canvas.onDivideLane(callbackFunc)
```

Lane 이 divide 되었을 때의 이벤트 리스너

Kind: instance method of [Canvas](#)

Param	Type	Description
callbackFunc	function	콜백함수(event, dividedLane)

```
canvas.onDrawLabel(callbackFunc)
```

라벨이 Draw 되었을 때의 이벤트 리스너

Kind: instance method of [Canvas](#)

Param	Type	Description
callbackFunc	function	콜백함수(event, shapeElement, labelText)

```
canvas.onLabelChanged(callbackFunc)
```

라벨이 Change 되었을 때의 이벤트 리스너

Kind: instance method of [Canvas](#)

Param	Type	Description
callbackFunc	function	콜백함수(event, shapeElement, afterText, beforeText)

```
canvas.onBeforeLabelChange(callbackFunc)
```

라벨이 Change 되기전 이벤트 리스너

Kind: instance method of [Canvas](#)

Param	Type	Description
callbackFunc	function	콜백함수(event, shapeElement, afterText, beforeText)

```
canvas.onRedrawShape(callbackFunc)
```

Shape 의 Redraw 되었을 때의 이벤트 리스너

Kind: instance method of [Canvas](#)

Param	Type	Description
callbackFunc	function	콜백함수(event, shapeElement)

```
canvas.onRemoveShape(callbackFunc)
```

Shape 이 Remove 될 때의 이벤트 리스너

Kind: instance method of [Canvas](#)

Param	Type	Description
callbackFunc	function	콜백함수(event, shapeElement)

```
canvas.onRotateShape(callbackFunc)
```

Shape 이 Rotate 될 때의 이벤트 리스너

Kind: instance method of [Canvas](#)

Param	Type	Description
callbackFunc	function	콜백함수(event, element, angle)

```
canvas.onMoveShape(callbackFunc)
```

Shape 이 Move 되었을 때의 이벤트 리스너

Kind: instance method of [Canvas](#)

Param	Type	Description
callbackFunc	function	콜백함수(event, shapeElement, offset)

```
canvas.onResizeShape(callbackFunc)
```

Shape 이 Resize 되었을 때의 이벤트 리스너

Kind: instance method of [Canvas](#)

Param	Type	Description
callbackFunc	function	콜백함수(event, shapeElement, offset)

canvas.onBeforeConnectShape(callbackFunc)

Shape 이| Connect 되기전 이벤트 리스너

Kind: instance method of [Canvas](#)

Param	Type	Description
callbackFunc	function	콜백함수(event, edgeElement, fromElement, toElement)

canvas.onBeforeRemoveShape(callbackFunc)

Shape 이| Remove 되기전 이벤트 리스너

Kind: instance method of [Canvas](#)

Param	Type	Description
callbackFunc	function	콜백함수(event, element)

canvas.onConnectShape(callbackFunc)

Shape 이| Connect 되었을 때의 이벤트 리스너

Kind: instance method of [Canvas](#)

Param	Type	Description
callbackFunc	function	콜백함수(event, edgeElement, fromElement, toElement)

canvas.onDisconnectShape(callbackFunc)

Shape 의 Disconnect 되었을 때의 이벤트 리스너

Kind: instance method of [Canvas](#)

Param	Type	Description
callbackFunc	function	콜백함수(event, edgeElement, fromElement, toElement)

canvas.onGroup(callbackFunc)

Shape 의 Grouping 되었을 때의 이벤트 리스너

Kind: instance method of [Canvas](#)

Param	Type	Description
callbackFunc	function	콜백함수(event, groupElement)

canvas.onUnGroup(callbackFunc)

Shape 의 UnGrouping 되었을 때의 이벤트 리스너

Kind: instance method of [Canvas](#)

Param	Type	Description
callbackFunc	function	콜백함수(event, ungroupedElements)

```
canvas.onCollapsed(callbackFunc)
```

Group 이 Collapse 되었을 때의 이벤트 리스너

Kind: instance method of [Canvas](#)

Param	Type	Description
callbackFunc	function	콜백함수(event, element)

```
canvas.onExpanded(callbackFunc)
```

Group 이 Expand 되었을 때의 이벤트 리스너

Kind: instance method of [Canvas](#)

Param	Type	Description
callbackFunc	function	콜백함수(event, element)

OG.handler : object

Kind: static namespace of [OG](#)

- [.handler](#) : object
  - [.EventHandler](#)
    - [new OG.handler.EventHandler\(renderer, config\)](#)
    - [.enableEditLabel\(element\)](#)
    - [.setMovable\(element, isMovable\)](#)
    - [.setConnectable\(element, guide, isConnectable\)](#)

- [.setResizable\(element, guide, isResizable\)](#)
- [.setClickSelectable\(element, isSelectable\)](#)
- [.setGroupDropable\(element\)](#)
- [.setDragSelectable\(isSelectable\)](#)
- [.setEnableHotKey\(isEnableHotKey\)](#)
- [.enableRootContextMenu\(\)](#)
- [.enableShapeContextMenu\(\)](#)
- [.selectShape\(element\)](#)
- [.selectShapes\(element\)](#)
- [.bringToFront\(\)](#)
- [.sendToBack\(\)](#)
- [.bringForward\(\)](#)
- [.sendBackward\(\)](#)
- [.deleteSelectedShape\(\)](#)
- [.changeShape\(\)](#)
- [.showProperty\(\)](#)
- [.selectAll\(\)](#)
- [.copySelectedShape\(\)](#)
- [.cutSelectedShape\(\)](#)
- [.pasteSelectedShape\(\)](#)
- [.duplicateSelectedShape\(\)](#)
- [.groupSelectedShape\(\)](#)
- [.ungroupSelectedShape\(\)](#)
- [.rotateSelectedShape\(angle\)](#)
- [.setLineWidthSelectedShape\(lineWidth\)](#)
- [.setLineColorSelectedShape\(lineColor\)](#)

- [.setLoopTypeSelectedShape\(lineType\)](#)
- [.setLineStyleSelectedShape\(lineStyle\)](#)
- [.setArrowStartSelectedShape\(arrowType\)](#)
- [.setArrowEndSelectedShape\(carrotType\)](#)
- [.setFillColorSelectedShape\(fillColor\)](#)
- [.setFontFamilySelectedShape\(fontFamily\)](#)
- [.setFontSizeSelectedShape\(fontSize\)](#)
- [.setFontColorSelectedShape\(fontColor\)](#)
- [.setFontWeightSelectedShape\(fontWeight\)](#)
- [.setFontStyleSelectedShape\(fontStyle\)](#)
- [.setTextDecorationSelectedShape\(textDecoration\)](#)
- [.setLabelDirectionSelectedShape\(labelDirection\)](#)
- [.setLabelAngleSelectedShape\(labelAngle\)](#)
- [.setLabelPositionSelectedShape\(labelPosition\)](#)
- [.setLabelVerticalSelectedShape\(verticalAlign\)](#)
- [.setLabelHorizontalSelectedShape\(horizontalAlign\)](#)
- [.setLabelSelectedShape\(label\)](#)
- [.setEdgeFromLabelSelectedShape\(label\)](#)
- [.setEdgeToLabelSelectedShape\(label\)](#)
- [.zoomIn\(\)](#)
- [.zoomOut\(\)](#)
- [.fitWindow\(\)](#)
- [.setConnectGuide\(element, isConnectable\)](#)

handler.EventHandler

Kind: static class of [handler](#)

Requires: module:OG.renderer.\*

Author: [Seungpil Park \(mailto:spark@uengine.org\)](mailto:Seungpil.Park@uengine.org)

- [.EventHandler](#)
  - [new OG.handler.EventHandler\(renderer, config\)](#)
  - [.enableEditLabel\(element\)](#)
  - [.setMovable\(element, isMovable\)](#)
  - [.setConnectable\(element, guide, isConnectable\)](#)
  - [.setResizable\(element, guide, isResizable\)](#)
  - [.setClickSelectable\(element, isSelectable\)](#)
  - [.setGroupDropable\(element\)](#)
  - [.setDragSelectable\(isSelectable\)](#)
  - [.enableHotKey\(isEnableHotKey\)](#)
  - [.enableRootContextMenu\(\)](#)
  - [.enableShapeContextMenu\(\)](#)
  - [.selectShape\(element\)](#)
  - [.selectShapes\(element\)](#)
  - [.bringToFront\(\)](#)
  - [.sendToBack\(\)](#)
  - [.bringForward\(\)](#)
  - [.sendBackward\(\)](#)

- [.deleteSelectedShape\(\)](#)
- [.changeShape\(\)](#)
- [.showProperty\(\)](#)
- [.selectAll\(\)](#)
- [.copySelectedShape\(\)](#)
- [.cutSelectedShape\(\)](#)
- [.pasteSelectedShape\(\)](#)
- [.duplicateSelectedShape\(\)](#)
- [.groupSelectedShape\(\)](#)
- [.ungroupSelectedShape\(\)](#)
- [.rotateSelectedShape\(angle\)](#)
- [.setLineWidthSelectedShape\(lineWidth\)](#)
- [.setLineColorSelectedShape\(lineColor\)](#)
- [.setLoopTypeSelectedShape\(lineType\)](#)
- [.setLineStyleSelectedShape\(lineStyle\)](#)
- [.setArrowStartSelectedShape\(arrowType\)](#)
- [.setArrowEndSelectedShape\(arrowType\)](#)
- [.setFillColorSelectedShape\(fillColor\)](#)
- [.setFontFamilySelectedShape\(fontFamily\)](#)
- [.setFontSizeSelectedShape\(fontSize\)](#)
- [.setFontColorSelectedShape\(fontColor\)](#)
- [.setFontWeightSelectedShape\(fontWeight\)](#)
- [.setFontStyleSelectedShape\(fontStyle\)](#)
- [.setTextDecorationSelectedShape\(textDecoration\)](#)
- [.setLabelDirectionSelectedShape\(labelDirection\)](#)
- [.setLabelAngleSelectedShape\(labelAngle\)](#)

- [.setLabelPositionSelectedShape\(labelPosition\)](#)
- [.setLabelVerticalSelectedShape\(verticalAlign\)](#)
- [.setLabelHorizontalSelectedShape\(horizontalAlign\)](#)
- [.setLabelSelectedShape\(label\)](#)
- [.setEdgeFromLabelSelectedShape\(label\)](#)
- [.setEdgeToLabelSelectedShape\(label\)](#)
- [.zoomIn\(\)](#)
- [.zoomOut\(\)](#)
- [.fitWindow\(\)](#)
- [.setConnectGuide\(element, isConnectable\)](#)

```
new OG.handler.EventHandler(renderer, config)
```

## Event Handler

Param	Type	Description
renderer	<a href="#">IRenderer</a>	렌더러
config	Object	Configuration

```
eventHandler.enableEditLabel(element)
```

주어진 Shape Element 의 라벨을 수정 가능하도록 한다.

Kind: instance method of [EventHandler](#)

Param	Type	Description
-------	------	-------------

```
element Element Shape Element
```

```
eventHandler.setMovable(element, isMovable)
```

Shape 엘리먼트의 이동 가능여부를 설정한다.

Kind: instance method of [EventHandler](#)

Param	Type	Description
-------	------	-------------

element	Element Shape	엘리먼트
---------	---------------	------

isMovable	Boolean	가능여부
-----------	---------	------

```
eventHandler.setConnectable(element, guide, isConnectable)
```

Shape 엘리먼트의 라인모양을 클릭하여 Shape 끼리 커넥트가 가능하게 한다.

Kind: instance method of [EventHandler](#)

Param	Type	Description
-------	------	-------------

element	Element Shape	엘리먼트
---------	---------------	------

guide	Object	JSON 포맷 가이드 정보
-------	--------	----------------

isConnectable	Boolean	가능여부
---------------	---------	------

```
eventHandler.setResizable(element, guide, isResizable)
```

Shape 엘리먼트의 리사이즈 가능여부를 설정한다.

Kind: instance method of [EventHandler](#)

Param	Type	Description
-------	------	-------------

element	Element Shape	엘리먼트
---------	---------------	------

guide Object JSON 포맷 가이드 정보  
isResizable Boolean 가능여부

```
eventHandler.setClickSelectable(element, isSelectable)
```

주어진 Shape Element 를 마우스 클릭하여 선택가능하도록 한다.

선택가능해야 리사이즈가 가능하다.

선택시 커넥트 모드일 경우 connect 가능하게 한다.

Kind: instance method of [EventHandler](#)

Param	Type	Description
element	Element Shape Element	
isSelectable	Boolean	선택가능여부

```
eventHandler.setGroupDropable(element)
```

Lane,Pool 엘리먼트가 새로 생성될 시 그룹을 맺도록 한다.

Kind: instance method of [EventHandler](#)

Param	Type	Description
element	Element Shape	엘리먼트

```
eventHandler.setDragSelectable(isSelectable)
```

마우스 드래그 영역지정 선택가능여부를 설정한다.

선택가능해야 리사이즈가 가능하다.

Kind: instance method of [EventHandler](#)

Param	Type	Description
isSelectable	Boolean	선택가능여부

```
eventHandler.setEditable(isSelectable)
```

HotKey 사용 가능여부를 설정한다. (Delete, Ctrl+A, Ctrl+C, Ctrl+V, Ctrl+G, Ctrl+U)

Kind: instance method of [EventHandler](#)

Param	Type	Description
isEnabledHotKey	Boolean	핫키가능여부

```
eventHandler.enableRootContextMenu()
```

캔버스에 마우스 우클릭 메뉴를 가능하게 한다.

Kind: instance method of [EventHandler](#)

```
eventHandler.enableShapeContextMenu()
```

Shape 에 마우스 우클릭 메뉴를 가능하게 한다.

Kind: instance method of [EventHandler](#)

```
eventHandler.selectShape(element)
```

주어진 Shape Element 를 선택된 상태로 되게 한다.

Kind: instance method of [EventHandler](#)

Param Type Description  
element Element Shape 엘리먼트

```
eventHandler.selectShapes(element)
```

주어진 다수의 Shape Element 를 선택된 상태로 되게 한다.

Kind: instance method of [EventHandler](#)

Param Type Description  
element Element Shape 엘리먼트

```
eventHandler.bringToFront()
```

메뉴 : 맨 앞으로 가져오기

Kind: instance method of [EventHandler](#)

```
eventHandler.sendToBack()
```

메뉴 : 맨 뒤로 보내기

Kind: instance method of [EventHandler](#)

```
eventHandler.bringForward()
```

메뉴 : 앞으로 가져오기

Kind: instance method of [EventHandler](#)

eventHandler.sendForward()

메뉴 : 뒤로 보내기

Kind: instance method of [EventHandler](#)

eventHandler.deleteSelectedShape()

메뉴 : 선택된 Shape 들을 삭제한다.

Kind: instance method of [EventHandler](#)

eventHandler.changeShape()

메뉴 : Shape를 선택한 모양으로 변경한다.

Kind: instance method of [EventHandler](#)

eventHandler.showProperty()

메뉴 : 속성 창 이벤트

Kind: instance method of [EventHandler](#)

eventHandler.selectAll()

메뉴 : 모든 Shape 들을 선택한다.

Kind: instance method of [EventHandler](#)

eventHandler.copySelectedShape()

메뉴 : 선택된 Shape 들을 복사한다.

Kind: instance method of [EventHandler](#)

eventHandler.cutSelectedShape()

메뉴 : 선택된 Shape 들을 잘라내기한다.

Kind: instance method of [EventHandler](#)

eventHandler.pasteSelectedShape()

메뉴 : 선택된 Shape 들을 붙여넣기 한다.

Kind: instance method of [EventHandler](#)

eventHandler.duplicateSelectedShape()

메뉴 : 선택된 Shape 들을 복제한다.

Kind: instance method of [EventHandler](#)

```
eventHandler.groupSelectedShape()
```

메뉴 : 선택된 Shape 들을 그룹핑한다.

Kind: instance method of [EventHandler](#)

```
eventHandler.ungroupSelectedShape()
```

메뉴 : 선택된 Shape 들을 그룹해제한다.

Kind: instance method of [EventHandler](#)

```
eventHandler.rotateSelectedShape(angle)
```

메뉴 : 선택된 Shape 들을 회전한다.

Kind: instance method of [EventHandler](#)

Param	Type	Description
angle	Number	회전각도

```
eventHandler.setWidthSelectedShape(lineWidth)
```

메뉴 : 선택된 Shape 들의 Line Width 를 설정한다.

Kind: instance method of [EventHandler](#)

Param Type  
lineWidth Number

```
eventHandler.setLineColorSelectedShape(lineColor)
```

메뉴 : 선택된 Shape 들의 Line Color 를 설정한다.

Kind: instance method of [EventHandler](#)

Param Type  
lineColor String

```
eventHandler.setLoopTypeSelectedShape(lineType)
```

메뉴 : 선택된 Shape 들의 Line Type 을 설정한다.

Kind: instance method of [EventHandler](#)

Param Type Description  
lineType String ['straight' 'plain' 'bezier']

```
eventHandler.setStyleSelectedShape(style)
```

메뉴 : 선택된 Shape 들의 Line Style 을 설정한다.

Kind: instance method of [EventHandler](#)

```
Param  Type  Description
lineStyle String ['-' '-' '-' '-' - '.' '.' -. ' ' - ' ' -- ' ' - . ' ' -- .' ' -- . ' ' ]
```

```
eventHandler.setArrowStartSelectedShape(arrowType)
```

메뉴 : 선택된 Edge Shape 들의 시작점 화살표 스타일을 설정한다.

Kind: instance method of [EventHandler](#)

```
Param  Type  Description
arrowType String ['block' 'open_block' 'classic' 'diamond' 'open_diamond' 'open' ' ']
```

```
eventHandler.setArrowEndSelectedShape(arrowType)
```

메뉴 : 선택된 Edge Shape 들의 끝점 화살표 스타일을 설정한다.

Kind: instance method of [EventHandler](#)

```
Param  Type  Description
arrowType String [] ['block' 'open_block' 'classic' 'diamond' 'open_diamond' 'open' ' ']
```

```
eventHandler.setFillColorSelectedShape(fillColor)
```

메뉴 : 선택된 Shape 들의 Fill Color 를 설정한다.

Kind: instance method of [EventHandler](#)

```
Param  Type
fillColor String
```

```
eventHandler.setFontFamilySelectedShape(fontFamily)
```

메뉴 : 선택된 Shape 들의 Font Family 를 설정한다.

Kind: instance method of [EventHandler](#)

Param	Type
fontFamily	String

```
eventHandler.setFontSizeSelectedShape(fontSize)
```

메뉴 : 선택된 Shape 들의 Font Size 를 설정한다.

Kind: instance method of [EventHandler](#)

Param	Type
fontSize	Number

```
eventHandler.setFontColorSelectedShape(fontColor)
```

메뉴 : 선택된 Shape 들의 Font Color 를 설정한다.

Kind: instance method of [EventHandler](#)

Param	Type
fontColor	String

```
eventHandler.setFontWeightSelectedShape(fontWeight)
```

메뉴 : 선택된 Shape 들의 Font Weight 를 설정한다.

Kind: instance method of [EventHandler](#)

Param Type Description  
fontWeight String ['bold' 'normal']

```
eventHandler.setFontStyleSelectedShape(fontStyle)
```

메뉴 : 선택된 Shape 들의 Font Style 을 설정한다.

Kind: instance method of [EventHandler](#)

Param Type Description  
fontStyle String ['italic' 'normal']

```
eventHandler.setTextDecorationSelectedShape(textDecoration)
```

메뉴 : 선택된 Shape 들의 Text Decoration 을 설정한다.

Kind: instance method of [EventHandler](#)

Param Type Description  
textDecoration String ['underline' 'none']

```
eventHandler.setLabelDirectionSelectedShape(labelDirection)
```

메뉴 : 선택된 Shape 들의 Label Direction 을 설정한다.

Kind: instance method of [EventHandler](#)

```
Param      Type Description
labelDirection String ['vertical' 'horizontal']
```

```
eventHandler.setLabelAngleSelectedShape(labelAngle)
```

메뉴 : 선택된 Shape 들의 Label Angle 을 설정한다.

Kind: instance method of [EventHandler](#)

```
Param      Type
labelAngle Number
```

```
eventHandler.setLabelPositionSelectedShape(labelPosition)
```

메뉴 : 선택된 Shape 들의 Label Position 을 설정한다.

Kind: instance method of [EventHandler](#)

```
Param      Type Description
labelPosition String ['top'      'bottom' 'left' 'right' 'center']
```

```
eventHandler.setLabelVerticalSelectedShape(verticalAlign)
```

메뉴 : 선택된 Shape 들의 라벨 Vertical Align 을 설정한다.

Kind: instance method of [EventHandler](#)

```
Param      Type Description
verticalAlign String ['top'      'middle' 'bottom']
```

```
eventHandler.setLabelHorizontalSelectedShape(horizontalAlign)
```

메뉴 : 선택된 Shape 들의 라벨 Horizontal Align 를 설정한다.

Kind: instance method of [EventHandler](#)

Param	Type	Description
horizontalAlign	String	['start'    'middle' 'end']

```
eventHandler.setLabelSelectedShape(label)
```

메뉴 : 선택된 Shape 의 라벨을 설정한다.

Kind: instance method of [EventHandler](#)

Param	Type
label	String

```
eventHandler.setEdgeFromLabelSelectedShape(label)
```

메뉴 : 선택된 Edge Shape 의 시작점 라벨을 설정한다.

Kind: instance method of [EventHandler](#)

Param	Type
label	String

```
eventHandler.setEdgeToLabelSelectedShape(label)
```

메뉴 : 선택된 Edge Shape 의 끝점 라벨을 설정한다.

Kind: instance method of [EventHandler](#)

Param Type

label String

eventHandler.zoomIn()

메뉴 : Zoom In

Kind: instance method of [EventHandler](#)

eventHandler.zoomOut()

메뉴 : Zoom Out

Kind: instance method of [EventHandler](#)

eventHandler.fitWindow()

메뉴 : 그려진 Shape 들을 캔버스 사이즈에 맞게 조절한다.

Kind: instance method of [EventHandler](#)

eventHandler.setConnectGuide(element, isConnectable)

Shape 엘리먼트의 setConnectGuide 에 관련된 이벤트

Kind: instance method of [EventHandler](#)

Param	Type	Description
element	Element Shape	엘리먼트
isConnectable	Boolean	가능여부

OG.layout : object

Kind: static namespace of [OG](#)

OG.renderer : object

Kind: static namespace of [OG](#)

- [.renderer](#) : object
  - [.IRenderer](#)
    - [new OG.renderer.IRenderer\(container, containerSize, backgroundColor, backgroundImage, config\)](#)
    - [.drawShape\(position, shape, size, style, id\)](#) ⇒ Element
    - [.drawGeom\(geometry, style\)](#) ⇒ Element
    - [.drawText\(position, text, size, style, id\)](#) ⇒ Element
    - [.drawImage\(position, imgSrc, size, style, id\)](#) ⇒ Element
    - [.drawEdge\(line, style, id, isSelf\)](#) ⇒ Element
    - [.drawLabel\(shapeElement, text, style\)](#) ⇒ Element
    - [.drawEdgeLabel\(shapeElement, text, type\)](#) ⇒ Element
    - [.redrawShape\(element, excludeEdgeId\)](#)

- [.redrawConnectedEdge\(element\)](#)
- [.connect\(fromTerminal, toTerminal, edge, style, label, preventTrigger\)](#) ⇒ Element
- [.disconnect\(element\)](#)
- [.drawDropOverGuide\(element\)](#)
- [.drawGuide\(element\)](#) ⇒ Object
- [.removeGuide\(element\)](#)
- [.removeAllGuide\(\)](#)
- [.removeConnectGuide\(element\)](#)
- [.removeAllConnectGuide\(\)](#)
- [.removeOtherConnectGuide\(element\)](#)
- [.removeAllVirtualEdge\(\)](#)
- [.drawEdgeGuide\(element\)](#) ⇒ Object
- [.drawRubberBand\(position, size, style\)](#) ⇒ Element
- [.removeRubberBand\(root\)](#)
- [.drawDraggableGuide\(element\)](#) ⇒ Element
- [.drawCollapseGuide\(element\)](#) ⇒ Element
- [.removeCollapseGuide\(element\)](#)
- [.group\(elements\)](#) ⇒ Element
- [.ungroup\(groupElements\)](#) ⇒ Array.<Element>
- [.addToGroup\(groupElement, elements\)](#)
- [.collapse\(element\)](#)
- [.expand\(element\)](#)
- [.clear\(\)](#)
- [.removeShape\(element\)](#)
- [.remove\(element\)](#)

- [.removeChild\(element\)](#)
- [.getRootElement\(\) ⇒ Element](#)
- [.getRootGroup\(\) ⇒ Element](#)
- [.getElementByPoint\(position\) ⇒ Element](#)
- [.getElementsByBBox\(envelope\) ⇒ Array.<Element>](#)
- [.setAttr\(element, attribute\)](#)
- [.getAttr\(element, attrName\) ⇒ Object](#)
- [.setShapeStyle\(element, style\)](#)
- [.toFront\(element\)](#)
- [.toBack\(element\)](#)
- [.bringForward\(element\)](#)
- [.sendBackward\(element\)](#)
- [.getCanvasSize\(\) ⇒ Array.<Number>](#)
- [.setCanvasSize\(size\)](#)
- [.fitCanvasSize\(minSize, fitScale\)](#)
- [.setViewBox\(position, size, isFit\)](#)
- [.getScale\(\) ⇒ Number](#)
- [.setScale\(scale\)](#)
- [.show\(element\)](#)
- [.hide\(element\)](#)
- [.appendChild\(srcElement, targetElement\) ⇒ Element](#)
- [.insertAfter\(srcElement, targetElement\) ⇒ Element](#)
- [.insertBefore\(srcElement, targetElement\) ⇒ Element](#)
- [.move\(element, offset\) ⇒ Element](#)
- [.moveCentroid\(element, position\) ⇒ Element](#)
- [.rotate\(element, angle\) ⇒ Element](#)

- `.resize(element, offset)` ⇒ Element
- `.resizeBox(element, size)` ⇒ Element
- `.clone(element)` ⇒ Element
- `.getElementById(id)` ⇒ Element
- `.getElementsByType(shapeType, excludeType)` ⇒ Array.  
`<Element>`
- `.getBBox(element)` ⇒ Object
- `.getRootBBox()` ⇒ Object
- `.getRealRootBBox()` ⇒ Object
- `.getContainer()` ⇒ Element
- `.isSVG()` ⇒ Boolean
- `.isVML()` ⇒ Boolean
- `.getPrevEdges(element)` ⇒ Array.`<Element>`
- `.getNextEdges(element)` ⇒ Array.`<Element>`
- `.getPrevShapes(element)` ⇒ Array.`<Element>`
- `.getPrevShapeIds(element)` ⇒ Array.`<String>`
- `.getNextShapes(element)` ⇒ Array.`<Element>`
- `.getNextShapeIds(element)` ⇒ Array.`<String>`
- `.getConnectGuideElements(Element)` ⇒ Array
- `.isTopGroup(Element)` ⇒ boolean
- `.getParent(element)` ⇒ Element
- `.getChilds(element)` ⇒ Array
- `.isGroup(element)` ⇒ boolean
- `.getAllShapes()` ⇒ Array
- `.getAllEdges()` ⇒ Array
- `.getAllNotEdges()` ⇒ Array
- `.isEdge()` ⇒ boolean

- `.isShape()` ⇒ boolean
- `.initHistory()`
- `.addHistory()`
- `.undo()`
- `.redo()`
- `.RaphaelRenderer` ← `IRenderer`
  - `new OG.renderer.RaphaelRenderer(container, containerSize, backgroundColor, backgroundImage, config)`
  - `.drawHtml(position, html, size, style, id)` ⇒ Element
  - `.getPointOfInflectionFromEdge()`
  - `.reconnect(edge)` ⇒ Element
  - `.disconnectOneWay(element, connectDirection)`
  - `.drawStickGuide(element, position)`
  - `.setTextListInController(element, textList)`
  - `.getTextListInController(element)`
  - `.getConnectGuideElements(Element)` ⇒ Array
  - `.getNotConnectGuideElements(Element)` ⇒ Array
  - `.removeConnectGuide(element)`
  - `.removeAllConnectGuide()`
  - `.removeOtherConnectGuide(element)`
  - `.getSpots(element)` ⇒ Array
  - `.getCircleSpots(element)` ⇒ Array
  - `.createVirtualSpot(x, x, element)` ⇒ Element
  - `.getVirtualSpot(element)` ⇒ Element
  - `.removeVirtualSpot(element)` ⇒ Element

- [.selectSpot\(선택한\)](#)
- [.getChildNodes\(element\)](#) ⇒ Array
- [.trimEdge\(element\)](#)
- [.trimConnectInnerVertice\(element\)](#) ⇒ Element
- [.trimConnectIntersection\(element\)](#) ⇒ Element
- [.getBoundary\(element\)](#) ⇒ Envelope
- [.setHighlight\(element, highlight\)](#)
- [.removeHighlight\(element, highlight\)](#)
- [.createTerminalString\(Element, point\)](#) ⇒ String
- [.createDefaultTerminalString\(Element\)](#) ⇒ String
- [.toFrontEdges\(\)](#)
- [.removeAllEdgeGuide\(\)](#)
- [.createVirtualEdge\(x, x, targetEle\)](#) ⇒ Element
- [.updateVirtualEdge\(x, x\)](#)
- [.getTargetfromVirtualEdge\(x, x\)](#)
- [.removeAllVirtualEdge\(\)](#)
- [.isLane\(Element\)](#) ⇒ boolean
- [.isPool\(Element\)](#) ⇒ boolean
- [.isScopeActivity\(Element\)](#) ⇒ boolean
- [.isHorizontalLane\(Element\)](#) ⇒ boolean
- [.isVerticalLane\(Element\)](#) ⇒ boolean
- [.isHorizontalPool\(Element\)](#) ⇒ boolean
- [.isVerticalPool\(Element\)](#) ⇒ boolean
- [.getChildLane\(Element\)](#) ⇒ Array
- [.enableDivideCount\(Element\)](#) ⇒ Number
- [.getExceptTitleLaneArea\(Element, boundary\)](#)

- [`.divideLane\(Element, quarterOrder\)`](#)
- [`.getBaseLanes\(Element\) ⇒ Array`](#)
- [`.getRootLane\(Element\) ⇒ Element`](#)
- [`.getIndexOfLane\(Element\) ⇒ Number`](#)
- [`.getDepthOfLane\(Element\) ⇒ Number`](#)
- [`.reEstablishLane\(Element\)`](#)
- [`.getBoundaryOfElements\(elements\) ⇒ Envelope`](#)
- [`.getNearestBaseLaneIndexAsDirection\(Element, direction\) ⇒ Number`](#)
- [`.getBoundaryOfInnerShapesGroup\(Element\) ⇒ Envelope`](#)
- [`.getSmallestBaseLane\(Element, baseLane\)`](#)
- [`.resizeLane\(Element, offset\)`](#)
- [`.removeLaneShape\(Element\)`](#)
- [`.getInnerShapesOfLane\(Element\)`](#)
- [`.fitLaneOrder\(Element\)`](#)
- [`.getRootGroupOfShape\(Element\) ⇒ Element`](#)
- [`.checkBridgeEdge\(Element\)`](#)
- [`.checkAllBridgeEdge\(\)`](#)
- [`.getInnerShapesOfGroup\(Element\)`](#)
- [`.getFrontForCoordinate\(point\) ⇒ Element`](#)
- [`.getFrontForBoundary\(boundary\) ⇒ Element`](#)
- [`.trimEdgeDirection\(Edge, FromShape, ToShape\) ⇒ Element`](#)
- [`.putInnerShapeToPool\(Element\) ⇒ Element`](#)
- [`.setDropablePool\(Element\) ⇒ Element`](#)
- [`.offDropablePool\(\)`](#)
- [`.drawShape\(position, shape, size, style, id\) ⇒ Element`](#)

- [.drawGeom\(geometry, style\)](#) ⇒ Element
- [.drawText\(position, text, size, style, id\)](#) ⇒ Element
- [.drawImage\(position, imgSrc, size, style, id\)](#) ⇒ Element
- [.drawEdge\(line, style, id, isSelf\)](#) ⇒ Element
- [.drawLabel\(shapeElement, text, style\)](#) ⇒ Element
  - [~getCenterOfEdge\(element\)](#) ⇒ OG.Coordinate
- [.drawEdgeLabel\(shapeElement, text, type\)](#) ⇒ Element
- [.redrawShape\(element, excludeEdgeId\)](#)
- [.redrawConnectedEdge\(element\)](#)
- [.connect\(fromTerminal, toTerminal, edge, style, label, preventTrigger\)](#) ⇒ Element
- [.disconnect\(element\)](#)
- [.drawDropOverGuide\(element\)](#)
- [.drawGuide\(element\)](#) ⇒ Object
- [.removeGuide\(element\)](#)
- [.removeAllGuide\(\)](#)
- [.drawEdgeGuide\(element\)](#) ⇒ Object
- [.drawRubberBand\(position, size, style\)](#) ⇒ Element
- [.removeRubberBand\(root\)](#)
- [.drawDraggableGuide\(element\)](#) ⇒ Element
- [.drawCollapseGuide\(element\)](#) ⇒ Element
- [.removeCollapseGuide\(element\)](#)
- [.group\(elements\)](#) ⇒ Element
- [.ungroup\(groupElements\)](#) ⇒ Array.<Element>
- [.addToGroup\(groupElement, elements\)](#)

- [.collapse\(element\)](#)
- [.expand\(element\)](#)
- [.clear\(\)](#)
- [.removeShape\(element\)](#)
- [.remove\(element\)](#)
- [.removeChild\(element\)](#)
- [.getRootElement\(\) ⇒ Element](#)
- [.getRootGroup\(\) ⇒ Element](#)
- [.getElementByPoint\(position\) ⇒ Element](#)
- [.getElementsByBBox\(envelope\) ⇒ Array.<Element>](#)
- [.setAttr\(element, attribute\)](#)
- [.getAttr\(element, attrName\) ⇒ Object](#)
- [.setShapeStyle\(element, style\)](#)
- [.toFront\(element\)](#)
- [.toBack\(element\)](#)
- [.bringForward\(element\)](#)
- [.sendBackward\(element\)](#)
- [.getCanvasSize\(\) ⇒ Array.<Number>](#)
- [.setCanvasSize\(size\)](#)
- [.fitCanvasSize\(minSize, fitScale\)](#)
- [.setViewBox\(position, size, isFit\)](#)
- [.getScale\(\) ⇒ Number](#)
- [.setScale\(scale\)](#)
- [.show\(element\)](#)
- [.hide\(element\)](#)
- [.appendChild\(srcElement, targetElement\) ⇒ Element](#)

- [.insertAfter\(srcElement, targetElement\)](#) ⇒ Element
- [.insertBefore\(srcElement, targetElement\)](#) ⇒ Element
- [.move\(element, offset\)](#) ⇒ Element
- [.moveCentroid\(element, position\)](#) ⇒ Element
- [.rotate\(element, angle\)](#) ⇒ Element
- [.resize\(element, offset\)](#) ⇒ Element
- [.resizeBox\(element, size\)](#) ⇒ Element
- [.clone\(element\)](#) ⇒ Element
- [.getElementById\(id\)](#) ⇒ Element
- [.getElementsByType\(shapeType, excludeType\)](#) ⇒ Array.<Element>
- [.getBBox\(element\)](#) ⇒ Object
- [.getRootBBox\(\)](#) ⇒ Object
- [.getRealRootBBox\(\)](#) ⇒ Object
- [.getContainer\(\)](#) ⇒ Element
- [.isSVG\(\)](#) ⇒ Boolean
- [.isVML\(\)](#) ⇒ Boolean
- [.getPrevEdges\(element\)](#) ⇒ Array.<Element>
- [.getNextEdges\(element\)](#) ⇒ Array.<Element>
- [.getPrevShapes\(element\)](#) ⇒ Array.<Element>
- [.getPrevShapeIds\(element\)](#) ⇒ Array.<String>
- [.getNextShapes\(element\)](#) ⇒ Array.<Element>
- [.getNextShapeIds\(element\)](#) ⇒ Array.<String>
- [.isTopGroup\(Element\)](#) ⇒ boolean
- [.getParent\(element\)](#) ⇒ Element
- [.getChilids\(element\)](#) ⇒ Array
- [.isGroup\(element\)](#) ⇒ boolean

- [.getAllShapes\(\)](#) ⇒ Array
- [.getAllEdges\(\)](#) ⇒ Array
- [.getAllNotEdges\(\)](#) ⇒ Array
- [.isEdge\(\)](#) ⇒ boolean
- [.isShape\(\)](#) ⇒ boolean
- [.initHistory\(\)](#)
- [.addHistory\(\)](#)
- [.undo\(\)](#)
- [.redo\(\)](#)

renderer.IRenderer

Kind: static class of [renderer](#)

Requires: module:OG.common.\* , module:OG.geometry.\* , module:OG.shape.\*

Author: [Seungpil Park \(mailto:spark@uengine.org\)](mailto:Seungpil.Park@uengine.org)

- [.IRenderer](#)

- [new OG.renderer.IRenderer\(container, containerSize, backgroundColor, backgroundImage, config\)](#)
- [.drawShape\(position, shape, size, style, id\)](#) ⇒ Element
- [.drawGeom\(geometry, style\)](#) ⇒ Element
- [.drawText\(position, text, size, style, id\)](#) ⇒ Element
- [.drawImage\(position, imgSrc, size, style, id\)](#) ⇒ Element

- [.drawEdge\(line, style, id, isSelf\)](#) ⇒ Element
- [.drawLabel\(shapeElement, text, style\)](#) ⇒ Element
- [.drawEdgeLabel\(shapeElement, text, type\)](#) ⇒ Element
- [.redrawShape\(element, excludeEdgeId\)](#)
- [.redrawConnectedEdge\(element\)](#)
- [.connect\(fromTerminal, toTerminal, edge, style, label, preventTrigger\)](#) ⇒ Element
- [.disconnect\(element\)](#)
- [.drawDropOverGuide\(element\)](#)
- [.drawGuide\(element\)](#) ⇒ Object
- [.removeGuide\(element\)](#)
- [.removeAllGuide\(\)](#)
- [.removeConnectGuide\(element\)](#)
- [.removeAllConnectGuide\(\)](#)
- [.removeOtherConnectGuide\(element\)](#)
- [.removeAllVirtualEdge\(\)](#)
- [.drawEdgeGuide\(element\)](#) ⇒ Object
- [.drawRubberBand\(position, size, style\)](#) ⇒ Element
- [.removeRubberBand\(root\)](#)
- [.drawDraggableGuide\(element\)](#) ⇒ Element
- [.drawCollapseGuide\(element\)](#) ⇒ Element
- [.removeCollapseGuide\(element\)](#)
- [.group\(elements\)](#) ⇒ Element
- [.ungroup\(groupElements\)](#) ⇒ Array.<Element>
- [.addToGroup\(groupElement, elements\)](#)
- [.collapse\(element\)](#)

- [.expand\(element\)](#)
- [.clear\(\)](#)
- [.removeShape\(element\)](#)
- [.remove\(element\)](#)
- [.removeChild\(element\)](#)
- [.getRootElement\(\) ⇒ Element](#)
- [.getRootGroup\(\) ⇒ Element](#)
- [.getElementByPoint\(position\) ⇒ Element](#)
- [.getElementsByBBox\(envelope\) ⇒ Array.<Element>](#)
- [.setAttr\(element, attribute\)](#)
- [.getAttr\(element, attrName\) ⇒ Object](#)
- [.setShapeStyle\(element, style\)](#)
- [.toFront\(element\)](#)
- [.toBack\(element\)](#)
- [.bringForward\(element\)](#)
- [.sendBackward\(element\)](#)
- [.getCanvasSize\(\) ⇒ Array.<Number>](#)
- [.setCanvasSize\(size\)](#)
- [.fitCanvasSize\(minSize, fitScale\)](#)
- [.setViewBox\(position, size, isFit\)](#)
- [.getScale\(\) ⇒ Number](#)
- [.setScale\(scale\)](#)
- [.show\(element\)](#)
- [.hide\(element\)](#)
- [.appendChild\(srcElement, targetElement\) ⇒ Element](#)
- [.insertAfter\(srcElement, targetElement\) ⇒ Element](#)

- `.insertBefore(srcElement, targetElement)` ⇒ Element
- `.move(element, offset)` ⇒ Element
- `.moveCentroid(element, position)` ⇒ Element
- `.rotate(element, angle)` ⇒ Element
- `.resize(element, offset)` ⇒ Element
- `.resizeBox(element, size)` ⇒ Element
- `.clone(element)` ⇒ Element
- `.getElementById(id)` ⇒ Element
- `.getElementsByType(shapeType, excludeType)` ⇒ Array.<Element>
- `.getBBox(element)` ⇒ Object
- `.getRootBBox()` ⇒ Object
- `.getRealRootBBox()` ⇒ Object
- `.getContainer()` ⇒ Element
- `.isSVG()` ⇒ Boolean
- `.isVML()` ⇒ Boolean
- `.getPrevEdges(element)` ⇒ Array.<Element>
- `.getNextEdges(element)` ⇒ Array.<Element>
- `.getPrevShapes(element)` ⇒ Array.<Element>
- `.getPrevShapeIds(element)` ⇒ Array.<String>
- `.getNextShapes(element)` ⇒ Array.<Element>
- `.getNextShapeIds(element)` ⇒ Array.<String>
- `.getConnectGuideElements(Element)` ⇒ Array
- `.isTopGroup(Element)` ⇒ boolean
- `.getParent(element)` ⇒ Element
- `.getchilds(element)` ⇒ Array
- `.isGroup(element)` ⇒ boolean

- [.getAllShapes\(\)](#) ⇒ Array
- [.getAllEdges\(\)](#) ⇒ Array
- [.getAllNotEdges\(\)](#) ⇒ Array
- [.isEdge\(\)](#) ⇒ boolean
- [.isShape\(\)](#) ⇒ boolean
- [.initHistory\(\)](#)
- [.addHistory\(\)](#)
- [.undo\(\)](#)
- [.redo\(\)](#)

```
new OG.renderer.IRenderer(container, containerSize, backgroundColor, backgroundImage, config)
```

도형의 Style 과 Shape 정보를 통해 캔버스에 렌더링 기능을 정의한 인터페이스

Param	Type	Description
container	HTMLElement   String	컨테이너 DOM element or ID
containerSize	Array.<Number>	컨테이너 Width, Height
backgroundColor	String	캔버스 배경색
backgroundImage	String	캔버스 배경이미지
config	Object	Configuration

```
iRenderer.drawShape(position, shape, size, style, id) ⇒ Element
```

Shape 을 캔버스에 위치 및 사이즈 지정하여 드로잉한다.

Kind: instance method of [IRenderer](#)

Returns: Element - Group DOM Element with geometry

Param	Type	Description
position	Array.<Number>	드로잉할 위치 좌표(중앙 기준)
shape	<a href="#">IShape</a>	Shape
size	Array.<Number>	Shape Width, Height
style	<a href="#">Style</a>   Object	스타일
id	String	Element ID 지정

### Example

```
renderer.drawShape([100, 100], new OG.CircleShape(), [50, 50], {stroke: 'red'});
```

iRenderer.drawGeom(geometry, style) ⇒ Element

Geometry 를 캔버스에 드로잉한다.

Kind: instance method of [IRenderer](#)

Returns: Element - Group DOM Element with geometry

Param	Type	Description
geometry	<a href="#">Geometry</a>	기하 객체
style	<a href="#">Style</a>   Object	스타일

iRenderer.drawText(position, text, size, style, id) ⇒ Element

Text 를 캔버스에 위치 및 사이즈 지정하여 드로잉한다.

(스타일 'text-anchor': 'start' or 'middle' or 'end' 에 따라 위치 기준이 다름)

Kind: instance method of [IRenderer](#)

Returns: Element - DOM Element

Param	Type	Description
position	Array.<Number>	드로잉할 위치 좌표(스타일 'text-anchor': 'start' or 'middle' or 'end'에 따라 기준이 다름)
text	String	텍스트
size	Array.<Number>	Text Width, Height, Angle
style	<a href="#">Style</a>   Object	스타일
id	String	Element ID 지정

### Example

```
renderer.drawText([100, 100], 'Hello', null, {'text-anchor': 'start'});
```

```
iRenderer.drawImage(position, imgSrc, size, style, id) => Element
```

Image 를 캔버스에 위치 및 사이즈 지정하여 드로잉한다.

Kind: instance method of [IRenderer](#)

Returns: Element - DOM Element

Param	Type	Description
position	Array.<Number>	드로잉할 위치 좌표(좌상단 기준)
imgSrc	String	이미지경로
size	Array.<Number>	Image Width, Height, Angle
style	<a href="#">Style</a>   Object	스타일
id	String	Element ID 지정

### Example

```
renderer.drawImage([100, 100], 'img.jpg', [50, 50]);
```

```
iRenderer.drawEdge(line, style, id, isSelf) => Element
```

라인을 캔버스에 드로잉한다.

Kind: instance method of [IRenderer](#)

Returns: Element - Group DOM Element with geometry

Param	Type	Description
line	<a href="#">Line</a>	라인
style	<a href="#">Style</a>   Object	스타일
id	String	Element ID 지정
isSelf	Boolean	셀프 연결 여부

iRenderer.drawLine(shapeElement, text, style) => Element

Shape 의 Label 을 캔버스에 위치 및 사이즈 지정하여 드로잉한다.

Kind: instance method of [IRenderer](#)

Returns: Element - DOM Element

Param	Type	Description
shapeElement	Element   String	Shape DOM element or ID
text	String	텍스트
style	Object	스타일

iRenderer.drawEdgeLabel(shapeElement, text, type) => Element

Edge 의 from, to Label 을 캔버스에 위치 및 사이즈 지정하여 드로잉한다.

Kind: instance method of [IRenderer](#)

Returns: Element - DOM Element

Param	Type	Description
-------	------	-------------

shapeElement	Element   String	Shape DOM element or ID
text	String	텍스트
type	String	유형(FROM or TO)

```
iRenderer.redrawShape(element, excludeEdgeId)
```

Element에 저장된 geom, angle, image, text 정보로 shape를 redraw 한다.

Kind: instance method of [IRenderer](#)

Param	Type	Description
element	Element	Shape 엘리먼트
excludeEdgeId	Array.<String>	redraw 제외할 Edge ID

```
iRenderer.redrawConnectedEdge(element)
```

Shape의 연결된 Edge를 redraw 한다.(이동 또는 리사이즈)

Kind: instance method of [IRenderer](#)

Param	Type
element	Element

```
iRenderer.connect(fromTerminal, toTerminal, edge, style, label, preventTrigger) ⇒ Element
```

두개의 터미널을 연결하고, 속성정보에 추가한다.

Kind: instance method of [IRenderer](#)

Returns: Element - 연결된 Edge 엘리먼트

Param	Type	Description
fromTerminal	Element   Array.<Number>	시작점 (fromTerminal)
toTerminal	Element   Array.<Number>	끝점 (toTerminal)
edge	Element	Edge Shape
style	<a href="#">Style</a>   Object	스타일
label	String	Label
preventTrigger	Boolean	이벤트 트리거 발생 막기

```
iRenderer.disconnect(element)
```

연결속성정보를 삭제한다. Edge 인 경우는 라인만 삭제하고, 일반 Shape 인 경우는 연결된 모든 Edge 를 삭제한다.

Kind: instance method of [IRenderer](#)

Param	Type
element	Element

```
iRenderer.drawDropOverGuide(element)
```

ID에 해당하는 Element 의 Edge 연결시 Drop Over 가이드를 드로잉한다.

Kind: instance method of [IRenderer](#)

Param	Type	Description
element	Element   String	Element 또는 ID

```
iRenderer.drawGuide(element) ⇒ Object
```

ID에 해당하는 Element 의 Move & Resize 용 가이드를 드로잉한다.

Kind: instance method of [IRenderer](#)

Param Type Description  
element Element | String Element 또는 ID

iRenderer.removeGuide(element)

ID에 해당하는 Element 의 Move & Resize 용 가이드를 제거한다.

Kind: instance method of [IRenderer](#)

Param Type Description  
element Element | String Element 또는 ID

iRenderer.removeAllGuide()

모든 Move & Resize 용 가이드를 제거한다.

Kind: instance method of [IRenderer](#)

iRenderer.removeConnectGuide(element)

ID에 해당하는 Element 의 Connect Guide 를 제거한다.

Kind: instance method of [IRenderer](#)

Param Type Description  
element Element | String Element 또는 ID

iRenderer.removeAllConnectGuide()

캔버스의 모든 Connect Guide 를 제거한다.

Kind: instance method of [IRenderer](#)

```
iRenderer.removeOtherConnectGuide(element)
```

ID에 해당하는 Element 이외의 모든 Connect Guide 를 제거한다.

Kind: instance method of [IRenderer](#)

Param	Type	Description
element	Element   String	Element 또는 ID

```
iRenderer.removeAllVirtualEdge()
```

캔버스의 가상선을 삭제한다.

Kind: instance method of [IRenderer](#)

```
iRenderer.drawEdgeGuide(element) => Object
```

ID에 해당하는 Edge Element 의 Move & Resize 용 가이드를 드로잉한다.

Kind: instance method of [IRenderer](#)

Param	Type	Description
element	Element   String	Element 또는 ID

```
iRenderer.drawRubberBand(position, size, style) => Element
```

Rectangle 모양의 마우스 드래그 선택 박스 영역을 드로잉한다.

Kind: instance method of [IRenderer](#)

Returns: Element - DOM Element

Param	Type	Description
position	Array.<Number>	드로잉할 위치 좌표(좌상단)
size	Array.<Number>	Text Width, Height, Angle
style	<a href="#">Style</a>   Object	스타일

```
iRenderer.removeRubberBand(root)
```

Rectangle 모양의 마우스 드래그 선택 박스 영역을 제거한다.

Kind: instance method of [IRenderer](#)

Param	Type	Description
root	Element	first, rubberBand 정보를 저장한 엘리먼트

```
iRenderer.drawDraggableGuide(element) => Element
```

ID에 해당하는 Element 의 Draggable 가이드를 드로잉한다.

Kind: instance method of [IRenderer](#)

Param	Type	Description
element	Element   String	Element 또는 ID

```
iRenderer.drawCollapseGuide(element) => Element
```

ID에 해당하는 Element 의 Collapse 가이드를 드로잉한다.

Kind: instance method of [IRenderer](#)

Param	Type	Description
element	Element   String	Element 또는 ID

```
iRenderer.removeCollapseGuide(element)
```

ID에 해당하는 Element 의 Collapse 가이드를 제거한다.

Kind: instance method of [IRenderer](#)

Param	Type
element	Element

```
iRenderer.group(elements) => Element
```

주어진 Shape 들을 그룹핑한다.

Kind: instance method of [IRenderer](#)

Returns: Element - Group Shape Element

Param	Type
elements	Array.<Element>

```
iRenderer.ungroup(groupElements) => Array.<Element>
```

주어진 그룹들을 그룹해제한다.

Kind: instance method of [IRenderer](#)

Returns: Array.<Element> - ungrouped Elements

Param	Type
groupElements	Array.<Element>

```
iRenderer.addToGroup(groupElement, elements)
```

주어진 Shape 들을 그룹에 추가한다.

Kind: instance method of [IRenderer](#)

Param	Type
groupElement	Element
elements	Array.<Element>

```
iRenderer.collapse(element)
```

주어진 Shape 이 그룹인 경우 collapse 한다.

Kind: instance method of [IRenderer](#)

Param	Type
element	Element

```
iRenderer.expand(element)
```

주어진 Shape 이 그룹인 경우 expand 한다.

Kind: instance method of [IRenderer](#)

Param Type  
element Element

iRenderer.clear()

드로잉된 모든 오브젝트를 클리어한다.

Kind: instance method of [IRenderer](#)

iRenderer.removeShape(element)

Shape 을 캔버스에서 관련된 모두를 삭제한다.

Kind: instance method of [IRenderer](#)

Param Type Description  
element Element | String Element 또는 ID

iRenderer.remove(element)

ID에 해당하는 Element 를 캔버스에서 제거한다.

Kind: instance method of [IRenderer](#)

Param Type Description  
element Element | String Element 또는 ID

```
iRenderer.removeChild(element)
```

하위 엘리먼트만 제거한다.

Kind: instance method of [IRenderer](#)

Param	Type	Description
element	Element   String	Element 또는 ID

```
iRenderer.getRootElement() => Element
```

랜더러 캔버스 Root Element 를 반환한다.

Kind: instance method of [IRenderer](#)

Returns: Element - Element

```
iRenderer.getRootGroup() => Element
```

랜더러 캔버스 Root Group Element 를 반환한다.

Kind: instance method of [IRenderer](#)

Returns: Element - Element

```
iRenderer.getElementByPoint(position) => Element
```

주어진 지점을 포함하는 Top Element 를 반환한다.

Kind: instance method of [IRenderer](#)

Returns: Element - Element

Param	Type	Description
positionArray.<Number>	위치 좌표	

iRenderer.getElementsByBBox(envelope) ⇒ Array.<Element>

주어진 Boundary Box 영역에 포함되는 Shape(GEOM, TEXT, IMAGE) Element 를 반환한다.

모든 vertices를 포함한 엘리먼트를 반환한다.

Kind: instance method of [IRenderer](#)

Returns: Array.<Element> - Element

Param	Type	Description
envelope	<a href="#">Envelope</a>	Boundary Box 영역

iRenderer.setAttr(element, attribute)

엘리먼트에 속성값을 설정한다.

Kind: instance method of [IRenderer](#)

Param	Type	Description
element	Element   String	Element 또는 ID
attributeObject		속성값

iRenderer.getAttr(element, attrName) ⇒ Object

엘리먼트 속성값을 반환한다.

Kind: instance method of [IRenderer](#)

Returns: Object - attribute 속성값

Param	Type	Description
element	Element   String	Element 또는 ID
attrName	String	속성 이름

```
iRenderer.setShapeStyle(element, style)
```

Shape 의 스타일을 변경한다.

Kind: instance method of [IRenderer](#)

Param	Type	Description
element	Element   String	Element 또는 ID
style	Object	스타일

```
iRenderer.toFront(element)
```

ID에 해당하는 Element 를 최상단 레이어로 이동한다.

Kind: instance method of [IRenderer](#)

Param	Type	Description
element	Element   String	Element 또는 ID

```
iRenderer.toBack(element)
```

ID에 해당하는 Element 를 최하단 레이어로 이동한다.

Kind: instance method of [IRenderer](#)

Param	Type	Description
element	Element   String	Element 또는 ID

```
iRenderer.bringForward(element)
```

ID에 해당하는 Element 를 앞으로 한단계 이동한다.

Kind: instance method of [IRenderer](#)

Param	Type	Description
element	Element   String	Element 또는 ID

```
iRenderer.sendBackward(element)
```

ID에 해당하는 Element 를 뒤로 한단계 이동한다.

Kind: instance method of [IRenderer](#)

Param	Type	Description
element	Element   String	Element 또는 ID

```
iRenderer.getCanvasSize() => Array.<Number>
```

랜더러 캔버스의 사이즈(Width, Height)를 반환한다.

Kind: instance method of [IRenderer](#)

Returns: Array.<Number> - Canvas Width, Height

```
iRenderer.setCanvasSize(size)
```

랜더러 캔버스의 사이즈(Width, Height)를 변경한다.

Kind: instance method of [IRenderer](#)

Param	Type	Description
size	Array.<Number>	Canvas Width, Height

```
iRenderer.fitCanvasSize(minSize, fitScale)
```

랜더러 캔버스의 사이즈(Width, Height)를 실제 존재하는 Shape 의 영역에 맞게 변경한다.

Kind: instance method of [IRenderer](#)

Param	Type	Description
minSize	Array.<Number>	Canvas 최소 Width, Height
fitScale	Boolean	주어진 minSize 에 맞게 fit 여부(Default:false)

```
iRenderer.setViewBox(position, size, isFit)
```

새로운 View Box 영역을 설정한다. (ZoomIn & ZoomOut 가능)

Kind: instance method of [IRenderer](#)

Param	Type	Description
position	Array.<Number>	위치 좌표(좌상단 기준)
size	Array.<Number>	Canvas Width, Height
isFit	Boolean	Fit 여부

iRenderer.getScale() ⇒ Number

Scale 을 반환한다. (리얼 사이즈 : Scale = 1)

Kind: instance method of [IRenderer](#)

Returns: Number - 스케일값

iRenderer.setScale(scale)

Scale 을 설정한다. (리얼 사이즈 : Scale = 1)

Kind: instance method of [IRenderer](#)

Param Type Description

scale Number 스케일값

iRenderer.show(element)

ID에 해당하는 Element 를 캔버스에서 show 한다.

Kind: instance method of [IRenderer](#)

Param Type Description

element Element | String Element 또는 ID

iRenderer.hide(element)

ID에 해당하는 Element 를 캔버스에서 hide 한다.

Kind: instance method of [IRenderer](#)

Param	Type	Description
element	Element   String	Element 또는 ID

```
iRenderer.appendChild(srcElement, targetElement) => Element
```

Source Element 를 Target Element 아래에 append 한다.

Kind: instance method of [IRenderer](#)

Returns: Element - Source Element

Param	Type	Description
srcElement	Element   String	Element 또는 ID
targetElement	Element   String	Element 또는 ID

```
iRenderer.insertAfter(srcElement, targetElement) => Element
```

Source Element 를 Target Element 이후에 insert 한다.

Kind: instance method of [IRenderer](#)

Returns: Element - Source Element

Param	Type	Description
srcElement	Element   String	Element 또는 ID
targetElement	Element   String	Element 또는 ID

```
iRenderer.insertBefore(srcElement, targetElement) => Element
```

Source Element 를 Target Element 이전에 insert 한다.

Kind: instance method of [IRenderer](#)

Returns: Element - Source Element

Param	Type	Description
srcElement	Element   String	Element 또는 ID
targetElement	Element   String	Element 또는 ID

iRenderer.move(element, offset) ⇒ Element

해당 Element 를 가로, 세로 Offset 만큼 이동한다.

Kind: instance method of [IRenderer](#)

Returns: Element - Element

Param	Type	Description
element	Element   String	Element 또는 ID
offset	Array.<Number>	[가로, 세로]

iRenderer.moveCentroid(element, position) ⇒ Element

주어진 중심좌표로 해당 Element 를 이동한다.

Kind: instance method of [IRenderer](#)

Returns: Element - Element

Param	Type	Description
element	Element   String	Element 또는 ID
positionArray	Array.<Number>	[x, y]

```
iRenderer.rotate(element, angle) => Element
```

중심 좌표를 기준으로 주어진 각도 만큼 회전한다.

Kind: instance method of [IRenderer](#)

Returns: Element - Element

Param	Type	Description
element	Element   String	Element 또는 ID
angle	Number	각도

```
iRenderer.resize(element, offset) => Element
```

상, 하, 좌, 우 외곽선을 이동한 만큼 리사이즈 한다.

Kind: instance method of [IRenderer](#)

Returns: Element - Element

Param	Type	Description
element	Element   String	Element 또는 ID
offset	Array.<Number>	[상, 하, 좌, 우] 각 방향으로 + 값

```
iRenderer.resizeBox(element, size) => Element
```

중심좌표는 고정한 채 Bounding Box 의 width, height 를 리사이즈 한다.

Kind: instance method of [IRenderer](#)

Returns: Element - Element

Param	Type	Description
element	Element   String	Element 또는 ID

size Array.<Number> [Width, Height]

iRenderer.clone(element) ⇒ Element

노드 Element 를 복사한다.

Kind: instance method of [IRenderer](#)

Returns: Element - Element

Param	Type	Description
element	Element   String	Element 또는 ID

iRenderer.getElementById(id) ⇒ Element

ID로 Node Element 를 반환한다.

Kind: instance method of [IRenderer](#)

Returns: Element - Element

Param	Type
id	String

iRenderer.getElementsByType(shapeType, excludeType) ⇒ Array.<Element>

Shape 타입에 해당하는 Node Element 들을 반환한다.

Kind: instance method of [IRenderer](#)

Returns: Array.<Element> - Element's Array

Param	Type	Description
shapeType	String	Shape 타입(GEOM, HTML, IMAGE, EDGE, GROUP), Null 이면 모든 타입
excludeType	String	제외 할 타입

iRenderer.getBBox(element) ⇒ Object

해당 엘리먼트의 BoundingBox 영역 정보를 반환한다.

Kind: instance method of [IRenderer](#)

Returns: Object - {width, height, x, y, x2, y2}

Param	Type
element	Element   String

iRenderer.getRootBBox() ⇒ Object

부모노드기준으로 캔버스 루트 엘리먼트의 BoundingBox 영역 정보를 반환한다.

Kind: instance method of [IRenderer](#)

Returns: Object - {width, height, x, y, x2, y2}

iRenderer.getRealRootBBox() ⇒ Object

부모노드기준으로 캔버스 루트 엘리먼트의 실제 Shape 이 차지하는 BoundingBox 영역 정보를 반환한다.

Kind: instance method of [IRenderer](#)

Returns: Object - {width, height, x, y, x2, y2}

```
iRenderer.getContainer() => Element
```

캔버스의 컨테이너 DOM element 를 반환한다.

Kind: instance method of [IRenderer](#)

Returns: Element - 컨테이너

```
iRenderer.isSVG() => Boolean
```

SVG 인지 여부를 반환한다.

Kind: instance method of [IRenderer](#)

Returns: Boolean - svg 여부

```
iRenderer.isVML() => Boolean
```

VML 인지 여부를 반환한다.

Kind: instance method of [IRenderer](#)

Returns: Boolean - vml 여부

```
iRenderer.getPrevEdges(element) => Array.<Element>
```

연결된 이전 Edge Element 들을 반환한다.

Kind: instance method of [IRenderer](#)

Returns: Array.<Element> - Previous Element's Array

Param	Type	Description
element	Element   String	Element 또는 ID

iRenderer.getNextEdges(element) ⇒ Array.<Element>

연결된 이후 Edge Element 들을 반환한다.

Kind: instance method of [IRenderer](#)

Returns: Array.<Element> - Previous Element's Array

Param	Type	Description
element	Element   String	Element 또는 ID

iRenderer.getPrevShapes(element) ⇒ Array.<Element>

연결된 이전 노드 Element 들을 반환한다.

Kind: instance method of [IRenderer](#)

Returns: Array.<Element> - Previous Element's Array

Param	Type	Description
element	Element   String	Element 또는 ID

iRenderer.getPrevShapeIds(element) ⇒ Array.<String>

연결된 이전 노드 Element ID들을 반환한다.

Kind: instance method of [IRenderer](#)

Returns: Array.<String> - Previous Element Id's Array

Param	Type	Description
element	Element   String	Element 또는 ID

```
iRenderer.getNextShapes(element) => Array.<Element>
```

연결된 이후 노드 Element 들을 반환한다.

Kind: instance method of [IRenderer](#)

Returns: Array.<Element> - Previous Element's Array

Param	Type	Description
element	Element   String	Element 또는 ID

```
iRenderer.getNextShapeIds(element) => Array.<String>
```

연결된 이후 노드 ID들을 반환한다.

Kind: instance method of [IRenderer](#)

Returns: Array.<String> - Previous Element Id's Array

Param	Type	Description
element	Element   String	Element 또는 ID

```
iRenderer.getConnectGuideElements(Element) => Array
```

Node 엘리먼트의 커넥트 가이드 엘리먼트를 반환한다.

Kind: instance method of [IRenderer](#)

Returns: Array - Array Element

Param Type Description  
Element Element 엘리먼트

iRenderer.isTopGroup(Element) ⇒ boolean

최상위 그룹 엘리먼트인지 반환한다.

Kind: instance method of [IRenderer](#)

Returns: boolean - true false

Param Type Description  
Element Element 엘리먼트

iRenderer.getParent(element) ⇒ Element

부모 엘리먼트를 반환한다. 부모가 루트일때는 반환하지 않는다.

Kind: instance method of [IRenderer](#)

Returns: Element - element 엘리먼트

Param Type Description  
element Element 엘리먼트

iRenderer.getChilds(element) ⇒ Array

그룹의 하위 엘리먼트를 반환한다.

Kind: instance method of [IRenderer](#)

Returns: Array - Elements

Param	Type	Description
element	Element	엘리먼트

iRenderer.isGroup(element) ⇒ boolean

그룹의 Shape 인지 반환한다. RootGroup 일 경우는 제외.

Kind: instance method of [IRenderer](#)

Returns: boolean - true false

Param	Type	Description
element	Element	엘리먼트

iRenderer.getAllShapes() ⇒ Array

캔버스의 모든 Shape 들을 리턴

Kind: instance method of [IRenderer](#)

Returns: Array - Elements

iRenderer.getAllEdges() ⇒ Array

캔버스의 모든 Edge를 리턴

Kind: instance method of [IRenderer](#)

Returns: Array - Edge Elements

iRenderer.getAllNotEdges() ⇒ Array

캔버스의 모든 Edge 가 아닌 shapes 를 리턴

Kind: instance method of [IRenderer](#)

Returns: Array - Edge Elements

iRenderer.isEdge() ⇒ boolean

Edge 여부를 판단.

Kind: instance method of [IRenderer](#)

Returns: boolean - true false

iRenderer.isShape() ⇒ boolean

Shape 여부를 판단.

Kind: instance method of [IRenderer](#)

Returns: boolean - true false

iRenderer.initHistory()

캔버스의 히스토리를 초기화한다.

Kind: instance method of [IRenderer](#)

iRenderer.addHistory()

캔버스에 히스토리를 추가한다.

Kind: instance method of [IRenderer](#)

iRenderer.undo()

캔버스의 Undo

Kind: instance method of [IRenderer](#)

iRenderer.redo()

캔버스의 Redo

Kind: instance method of [IRenderer](#)

renderer.RaphaelRenderer ⇐ [IRenderer](#)

Kind: static class of [renderer](#)

Extends: [IRenderer](#)

Requires: module:OG.common.\* , module:OG.geometry.\* , module:OG.shape.\* ,

module:raphael-2.1.0

Author: [Seungpil Park \(mailto:spark@uengine.org\)](mailto:Seungpil.Park@sparc.uengine.org)

- [.RaphaelRenderer](#) ⇐ [IRenderer](#)
  - [new OG.renderer.RaphaelRenderer\(container, containerSize, backgroundColor, backgroundImage, config\)](#)
  - [.drawHtml\(position, html, size, style, id\)](#) ⇒ Element
  - [.getPointOfInflectionFromEdge\(\)](#)
  - [.reconnect\(edge\)](#) ⇒ Element
  - [.disconnectOneWay\(element, connectDirection\)](#)
  - [.drawStickGuide\(element, position\)](#)
  - [.setTextListInController\(element, textList\)](#)
  - [.getTextListInController\(element\)](#)
  - [.getConnectGuideElements\(Element\)](#) ⇒ Array
  - [.getNotConnectGuideElements\(Element\)](#) ⇒ Array
  - [.removeConnectGuide\(element\)](#)
  - [.removeAllConnectGuide\(\)](#)
  - [.removeOtherConnectGuide\(element\)](#)
  - [.getSpots\(element\)](#) ⇒ Array
  - [.getCircleSpots\(element\)](#) ⇒ Array
  - [.createVirtualSpot\(x, x, element\)](#) ⇒ Element
  - [.getVirtualSpot\(element\)](#) ⇒ Element
  - [.removeVirtualSpot\(element\)](#) ⇒ Element
  - [.selectSpot\(선택한\)](#)
  - [.getChildNodes\(element\)](#) ⇒ Array
  - [.trimEdge\(element\)](#)

- [.trimConnectInnerVertice\(element\)](#) ⇒ Element
- [.trimConnectIntersection\(element\)](#) ⇒ Element
- [.getBoundary\(element\)](#) ⇒ Envelope
- [.setHighlight\(element, highlight\)](#)
- [.removeHighlight\(element, highlight\)](#)
- [.createTerminalString\(Element, point\)](#) ⇒ String
- [.createDefaultTerminalString\(Element\)](#) ⇒ String
- [.toFrontEdges\(\)](#)
- [.removeAllEdgeGuide\(\)](#)
- [.createVirtualEdge\(x, x, targetEle\)](#) ⇒ Element
- [.updateVirtualEdge\(x, x\)](#)
- [.getTargetfromVirtualEdge\(x, x\)](#)
- [.removeAllVirtualEdge\(\)](#)
- [.isLane\(Element\)](#) ⇒ boolean
- [.isPool\(Element\)](#) ⇒ boolean
- [.isScopeActivity\(Element\)](#) ⇒ boolean
- [.isHorizontalLane\(Element\)](#) ⇒ boolean
- [.isVerticalLane\(Element\)](#) ⇒ boolean
- [.isHorizontalPool\(Element\)](#) ⇒ boolean
- [.isVerticalPool\(Element\)](#) ⇒ boolean
- [.getChildLane\(Element\)](#) ⇒ Array
- [.enableDivideCount\(Element\)](#) ⇒ Number
- [.getExceptTitleLaneArea\(Element, boundary\)](#)
- [.divideLane\(Element, quarterOrder\)](#)
- [.getBaseLanes\(Element\)](#) ⇒ Array
- [.getRootLane\(Element\)](#) ⇒ Element

- [.getIndexOfLane\(Element\)](#) ⇒ Number
- [.getDepthOfLane\(Element\)](#) ⇒ Number
- [.reEstablishLane\(Element\)](#)
- [.getBoundaryOfElements\(elements\)](#) ⇒ [Envelope](#)
- [.getNearestBaseLaneIndexAsDirection\(Element, direction\)](#) ⇒ Number
- [.getBoundaryOfInnerShapesGroup\(Element\)](#) ⇒ [Envelope](#)
- [.getSmallestBaseLane\(Element, baseLane\)](#)
- [.resizeLane\(Element, offset\)](#)
- [.removeLaneShape\(Element\)](#)
- [.getInnerShapesOfLane\(Element\)](#)
- [.fitLaneOrder\(Element\)](#)
- [.getRootGroupOfShape\(Element\)](#) ⇒ Element
- [.checkBridgeEdge\(Element\)](#)
- [.checkAllBridgeEdge\(\)](#)
- [.getInnerShapesOfGroup\(Element\)](#)
- [.getFrontForCoordinate\(point\)](#) ⇒ Element
- [.getFrontForBoundary\(boundary\)](#) ⇒ Element
- [.trimEdgeDirection\(Edge, FromShape, ToShape\)](#) ⇒ Element
- [.putInnerShapeToPool\(Element\)](#) ⇒ Element
- [.setDropablePool\(Element\)](#) ⇒ Element
- [.offDropablePool\(\)](#)
- [.drawShape\(position, shape, size, style, id\)](#) ⇒ Element
- [.drawGeom\(geometry, style\)](#) ⇒ Element
- [.drawText\(position, text, size, style, id\)](#) ⇒ Element
- [.drawImage\(position, imgSrc, size, style, id\)](#) ⇒ Element
- [.drawEdge\(line, style, id, isSelf\)](#) ⇒ Element

- `.drawLabel(shapeElement, text, style)` ⇒ Element
  - `~getCenterOfEdge(element)` ⇒ OG.Coordinate
- `.drawEdgeLabel(shapeElement, text, type)` ⇒ Element
- `.redrawShape(element, excludeEdgeId)`
- `.redrawConnectedEdge(element)`
- `.connect(fromTerminal, toTerminal, edge, style, label, preventTrigger)` ⇒ Element
- `.disconnect(element)`
- `.drawDropOverGuide(element)`
- `.drawGuide(element)` ⇒ Object
- `.removeGuide(element)`
- `.removeAllGuide()`
- `.drawEdgeGuide(element)` ⇒ Object
- `.drawRubberBand(position, size, style)` ⇒ Element
- `.removeRubberBand(root)`
- `.drawDraggableGuide(element)` ⇒ Element
- `.drawCollapseGuide(element)` ⇒ Element
- `.removeCollapseGuide(element)`
- `.group(elements)` ⇒ Element
- `.ungroup(groupElements)` ⇒ Array.<Element>
- `.addToGroup(groupElement, elements)`
- `.collapse(element)`
- `.expand(element)`
- `.clear()`
- `.removeShape(element)`

- [.remove\(element\)](#)
- [.removeChild\(element\)](#)
- [.getRootElement\(\) ⇒ Element](#)
- [.getRootGroup\(\) ⇒ Element](#)
- [.getElementByPoint\(position\) ⇒ Element](#)
- [.getElementsByBBox\(envelope\) ⇒ Array.<Element>](#)
- [.setAttr\(element, attribute\)](#)
- [.getAttr\(element, attrName\) ⇒ Object](#)
- [.setShapeStyle\(element, style\)](#)
- [.toFront\(element\)](#)
- [.toBack\(element\)](#)
- [.bringForward\(element\)](#)
- [.sendBackward\(element\)](#)
- [.getCanvasSize\(\) ⇒ Array.<Number>](#)
- [.setCanvasSize\(size\)](#)
- [.fitCanvasSize\(minSize, fitScale\)](#)
- [.setViewBox\(position, size, isFit\)](#)
- [.getScale\(\) ⇒ Number](#)
- [.setScale\(scale\)](#)
- [.show\(element\)](#)
- [.hide\(element\)](#)
- [.appendChild\(srcElement, targetElement\) ⇒ Element](#)
- [.insertAfter\(srcElement, targetElement\) ⇒ Element](#)
- [.insertBefore\(srcElement, targetElement\) ⇒ Element](#)
- [.move\(element, offset\) ⇒ Element](#)
- [.moveCentroid\(element, position\) ⇒ Element](#)

- `.rotate(element, angle)` ⇒ Element
- `.resize(element, offset)` ⇒ Element
- `.resizeBox(element, size)` ⇒ Element
- `.clone(element)` ⇒ Element
- `.getElementById(id)` ⇒ Element
- `.getElementsByType(shapeType, excludeType)` ⇒ Array.<Element>
- `.getBBox(element)` ⇒ Object
- `.getRootBBox()` ⇒ Object
- `.getRealRootBBox()` ⇒ Object
- `.getContainer()` ⇒ Element
- `.isSVG()` ⇒ Boolean
- `.isVML()` ⇒ Boolean
- `.getPrevEdges(element)` ⇒ Array.<Element>
- `.getNextEdges(element)` ⇒ Array.<Element>
- `.getPrevShapes(element)` ⇒ Array.<Element>
- `.getPrevShapeIds(element)` ⇒ Array.<String>
- `.getNextShapes(element)` ⇒ Array.<Element>
- `.getNextShapeIds(element)` ⇒ Array.<String>
- `.isTopGroup(Element)` ⇒ boolean
- `.getParent(element)` ⇒ Element
- `.getchilds(element)` ⇒ Array
- `.isGroup(element)` ⇒ boolean
- `.getAllShapes()` ⇒ Array
- `.getAllEdges()` ⇒ Array
- `.getAllNotEdges()` ⇒ Array
- `.isEdge()` ⇒ boolean

- [.isShape\(\)](#) ⇒ boolean
- [.initHistory\(\)](#)
- [.addHistory\(\)](#)
- [.undo\(\)](#)
- [.redo\(\)](#)

```
new OG.renderer.RaphaelRenderer(container, containerSize, backgroundColor, backgroundImage, config)
```

Raphael 라이브러리를 이용하여 구현한 랜더러 캔버스 클래스

- 노드에 추가되는 속성 : \_type, \_shape, \_selected, \_from, \_to, \_fromedge, \_toedge
- 노드에 저장되는 값 : shape : { geom, angle, image, text }, data : 커스텀 Object

Param	Type	Description
container	HTMLElement   String	컨테이너 DOM element or ID
containerSize	Array.<Number>	컨테이너 Width, Height
backgroundColor	String	캔버스 배경색
backgroundImage	String	캔버스 배경이미지
config	Object	Configuration

```
raphaelRenderer.drawHtml(position, html, size, style, id) ⇒ Element
```

임베드 HTML String 을 캔버스에 위치 및 사이즈 지정하여 드로잉한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: Element - DOM Element

Param	Type	Description
position	Array.<Number>	드로잉할 위치 좌표(중앙 기준)
html	String	임베드 HTML String
size	Array.<Number>	Image Width, Height, Angle
style	<a href="#">Style</a>   Object	스타일
id	String	Element ID 지정

raphaelRenderer.\_getPointOfInflectionFromEdge()

사용자가 지정한 변곡점을 반환

- 조건:

오직 4개의 점을 가지고있는 다각선에서

변곡점이 자동으로 그려진 점의 위치가 아닐 때

- issue: 사용자가 지정한 변곡점(?)은 유지되어야 함

Kind: instance method of [RaphaelRenderer](#)

raphaelRenderer.reconnect(edge) ⇒ Element

연결된 터미널의 vertices 를 초기화하여 재연결한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: Element - 연결된 Edge 엘리먼트

Param	Type	Description
edge	Element	Edge Shape

```
raphaelRenderer.disconnectOneWay(element, connectDirection)
```

단방향 연결속성정보를 삭제한다. Edge 인 경우에만 해당한다.

Kind: instance method of [RaphaelRenderer](#)

Param	Type	Description
element	Element	
connectDirection	String	연결방향 'from' or 'to'

```
raphaelRenderer.drawStickGuide(element, position)
```

ID에 해당하는 Element 의 Stick 용 가이드를 드로잉한다.

Kind: instance method of [RaphaelRenderer](#)

Param	Type	Description
element	Element   String	Element 또는 ID
position	Object	

```
raphaelRenderer.setTextListInController(element, textList)
```

Shape 의 선 연결 커스텀 컨트롤러를 설정한다.

Kind: instance method of [RaphaelRenderer](#)

Param	Type	Description

```
element Element | String Element 또는 ID  
textList Array 텍스트 리스트
```

```
raphaelRenderer.getTextListInController(element)
```

Shape 의 선 연결 커스텀 컨트롤러를 가져온다.

Kind: instance method of [RaphaelRenderer](#)

Param	Type	Description
element	Element   String Element 또는 ID	

```
raphaelRenderer.getConnectGuideElements(Element) => Array
```

Node 엘리먼트의 커넥트 가이드 영역 엘리먼트를 반환한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [getConnectGuideElements](#)

Returns: Array - Array Element

Param	Type	Description
Element	Element	엘리먼트

```
raphaelRenderer.getNotConnectGuideElements(Element) => Array
```

Node 엘리먼트의 커넥트 가이드 영역을 제외한 엘리먼트를 반환한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: Array - Array Element

Param Type Description

Element Element 엘리먼트

```
raphaelRenderer.removeConnectGuide(element)
```

ID에 해당하는 Element 의 Connect Guide 를 제거한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [removeConnectGuide](#)

Param Type Description

element Element | String Element 또는 ID

```
raphaelRenderer.removeAllConnectGuide()
```

캔버스의 모든 Connect Guide 를 제거한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [removeAllConnectGuide](#)

```
raphaelRenderer.removeOtherConnectGuide(element)
```

ID에 해당하는 Element 이외의 모든 Connect Guide 를 제거한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [removeOtherConnectGuide](#)

Param Type Description

element Element | String Element 또는 ID

raphaelRenderer.getSpots(element) => Array

Element 내부의 Spot 들을 반환한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: Array - Spot Element Array

Param Type Description

element Element | String Element 또는 ID

raphaelRenderer.getCircleSpots(element) => Array

Element 내부의 변곡점 Spot 들만 반환한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: Array - Spot Element Array

Param Type Description

element Element | String Element 또는 ID

raphaelRenderer.createVirtualSpot(x, y, element) => Element

주어진 좌표와 가장 Edge Element의 가장 가까운 거리에 가상 변곡점 스팟을 생성한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: Element - Spot Element

Param	Type	Description
x	Number	이벤트의 캔버스 기준 x 좌표
x	Number	이벤트의 캔버스 기준 y 좌표
element	Element   String	Element 또는 ID

```
raphaelRenderer.getVirtualSpot(element) => Element
```

Element 내부의 가상 변곡점 스팟을 반환한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: Element - Spot Element

Param	Type	Description
element	Element   String	Element 또는 ID

```
raphaelRenderer.removeVirtualSpot(element) => Element
```

Element 내부의 가상 변곡점 스팟을 삭제한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: Element - Spot Element

Param	Type	Description
element	Element   String	Element 또는 ID

```
raphaelRenderer.selectSpot(선택한)
```

Element 내부의 Spot 중 선택한 스팟을 제외하고 모두 삭제하고, 가이드라인도 삭제한다.

Kind: instance method of [RaphaelRenderer](#)

Param	Type	Description
	선택한 Element   String	spot Element 또는 ID

raphaelRenderer.getChildNodes(element) ⇒ Array

하위 엘리먼트들을 반환한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: Array - Array Element

Param	Type	Description
	element Element   String	Element 또는 ID

raphaelRenderer.trimEdge(element)

Edge Element 내부의 패스중 나열된 두 꼭지점이 매우 짧은 선일 경우 하나의 꼭지점으로 정리한다.

Edge Element 내부의 패스중 나열된 세 꼭지점이 평행에 가까울 경우 하나의 선분으로 정리한다.

Kind: instance method of [RaphaelRenderer](#)

Param	Type	Description
	element Element   String	Element 또는 ID

raphaelRenderer.trimConnectInnerVertice(element) ⇒ Element

Edge Element의 연결 정보가 있을 경우 연결대상과 꼭지점의 다중 중복을 정리한다.

다중 중복 정리 후 Edge 의 모양이 직선인 경우 새로운 plain 을 제작한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: Element - element

Param Type Description

element Element | String Element 또는 ID

raphaelRenderer.trimConnectIntersection(element) => Element

Edge Element의 연결 정보가 있을 경우 선분과 연결대상의 연결점을 자연스럽게 한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: Element - element

Param Type Description

element Element | String Element 또는 ID

raphaelRenderer.getBoundary(element) => [Envelope](#)

ID에 해당하는 Element 의 바운더리 영역을 리턴한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: [Envelope](#) - Envelope 영역

Param Type Description

element Element | String Element 또는 ID

raphaelRenderer.setHighlight(element, highlight)

Element 에 하이라이트 속성을 부여한다.

Kind: instance method of [RaphaelRenderer](#)

Param	Type	Description
element	Element   String	Element 또는 ID
highlightObject		HIGHLIGHT 속성 집합.

```
raphaelRenderer.removeHighlight(element, highlight)
```

Element에 하이라이트 속성을 제거한다.

Kind: instance method of [RaphaelRenderer](#)

Param	Type	Description
element	Element   String	Element 또는 ID
highlightObject		HIGHLIGHT 속성 집합.

```
raphaelRenderer.createTerminalString(Element, point) => String
```

터미널 문자열을 생성한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: String - terminal 터미널 문자열

Param	Type	Description
Element	Element   String	Element 또는 ID
point	Array	연결 좌표정보 [x,y]

```
raphaelRenderer.createDefaultTerminalString(Element) => String
```

디폴트 터미널 문자열을 생성한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: String - terminal 터미널 문자열

Param	Type	Description
Element	Element   String	Element 또는 ID

```
raphaelRenderer.toFrontEdges()
```

캔버스의 Edge 들을 항상 최상단으로 이동시킨다.

Kind: instance method of [RaphaelRenderer](#)

```
raphaelRenderer.removeAllEdgeGuide()
```

캔버스의 Edge 들의 가이드를 제거한다.

Kind: instance method of [RaphaelRenderer](#)

```
raphaelRenderer.createVirtualEdge(x, x, targetEle) => Element
```

주어진 좌표와 선택된 Element 사이에 가상 연결선을 생성한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: Element - Edge Element

Param	Type	Description
x	Number	이벤트의 캔버스 기준 x 좌표
x	Number	이벤트의 캔버스 기준 y 좌표
targetEle	Element   String	Element 또는 ID

```
raphaelRenderer.updateVirtualEdge(x, x)
```

캔버스의 가상 연결선을 업데이트한다.

Kind: instance method of [RaphaelRenderer](#)

Param	Type	Description
x	Number	이벤트의 캔버스 기준 x 좌표
x	Number	이벤트의 캔버스 기준 y 좌표

```
raphaelRenderer.getTargetFromVirtualEdge(x, x)
```

캔버스의 가상선의 타겟 엘리먼트를 구한다.

Kind: instance method of [RaphaelRenderer](#)

Param	Type	Description
x	Number	이벤트의 캔버스 기준 x 좌표
x	Number	이벤트의 캔버스 기준 y 좌표

```
raphaelRenderer.removeAllVirtualEdge()
```

캔버스의 가상선을 삭제한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [removeAllVirtualEdge](#)

```
raphaelRenderer.isLane(Element) => boolean
```

도형의 Lane 타입 여부를 판별한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: boolean - true false

Param	Type	Description
Element	Element   String	Element 또는 ID

```
raphaelRenderer.isPool(Element) => boolean
```

도형의 Pool 타입 여부를 판별한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: boolean - true false

Param	Type	Description
Element	Element   String	Element 또는 ID

```
raphaelRenderer.isScopeActivity(Element) => boolean
```

도형의 ScopeActivity 타입 여부를 판별한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: boolean - true false

Param	Type	Description
Element	Element   String	Element 또는 ID

```
raphaelRenderer.isHorizontalLane(Element) => boolean
```

도형의 HorizontalLaneShape 타입 여부를 판별한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: boolean - true false

Param	Type	Description
Element	Element   String	Element 또는 ID

```
raphaelRenderer.isVerticalLane(Element) => boolean
```

도형의 VerticalLaneShape 타입 여부를 판별한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: boolean - true false

Param	Type	Description
Element	Element   String	Element 또는 ID

```
raphaelRenderer.isHorizontalPool(Element) => boolean
```

도형의 HorizontalPoolShape 타입 여부를 판별한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: boolean - true false

Param	Type	Description
Element	Element   String	Element 또는 ID

```
raphaelRenderer.isVerticalPool(Element) => boolean
```

도형의 VerticalPoolShape 타입 여부를 판별한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: boolean - true false

Param	Type	Description
Element	Element   String	Element 또는 ID

```
raphaelRenderer.getChildLane(Element) => Array
```

Lane 타입 도형 하위의 Lane 타입들을 리턴한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: Array - childsLanes

Param	Type	Description
Element	Element   String	Element 또는 ID

```
raphaelRenderer.enableDivideCount(Element) => Number
```

Lane 타입이 내부적으로 분기가 가능한 수를 리턴한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: Number - 0,1,2

Param	Type	Description
Element	Element   String	Element 또는 ID

```
raphaelRenderer.getExceptTitleLaneArea(Element, boundary)
```

Lane, Pool 의 타이틀 영역을 제외한 boundary 를 리턴한다.

Kind: instance method of [RaphaelRenderer](#)

Param	Type	Description
Element	Element   String	Element 또는 ID
boundary	Envelope	

```
raphaelRenderer.divideLane(Element, quarterOrder)
```

Lane 을 분기한다.

Kind: instance method of [RaphaelRenderer](#)

Param	Type	Description
Element	Element   String	Element 또는 ID
quarterOrder	String	분기 명령 QUARTER_UPPER
		QUARTER_LOW QUARTER_BISECTOR QUARTER_THIRDS

```
raphaelRenderer.getBaseLanes(Element) => Array
```

Lane 의 최상의 Lane 으로부터 모든 Base Lane 들을 반환한다.

Base Lane 은 자식 Lane 을 가지지 않는 Lane 을 뜻함.

반환하는 Array 는 좌표상의 값을 기준으로 정렬되어 있는 상태이다.

Kind: instance method of [RaphaelRenderer](#)

Returns: Array - childBaseLanes

Param	Type	Description
Element	Element   String	Element 또는 ID

```
raphaelRenderer.getRootLane(Element) => Element
```

Lane 의 최상위 Lane 을 반환한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: Element - Lane Element

Param	Type	Description
Element	Element   String	Element 또는 ID

```
raphaelRenderer.getIndexOfLane(Element) => Number
```

Lane 의 BaseLane 으로부터 자신의 순서를 구한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: Number - index

Param	Type	Description
Element	Element   String	Element 또는 ID

```
raphaelRenderer.getDepthOfLane(Element) => Number
```

Lane 의 최상위 Lane 으로부터 Depth를 구한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: Number - depth

Param	Type	Description
-------	------	-------------

Element Element | String Element 또는 ID

```
raphaelRenderer.reEstablishLane(Element)
```

Lane 의 BaseLane 영역을 기준으로 전체 Lane 의 구조를 재정립한다.

Kind: instance method of [RaphaelRenderer](#)

Param Type Description

Element Element | String Element 또는 ID

```
raphaelRenderer.getBoundaryOfElements(elements) => Envelope
```

주어진 Shape 들의 바운더리 영역을 반환한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: [Envelope](#) - Envelope 영역

Param Type

elements Array.<Element>

```
raphaelRenderer.getNearestBaseLaneIndexAsDirection(Element, direction) => Number
```

Lane 의 baseLane 들 중

Lane의 주어진 direction 과 BaseLane 의 주어진 direction 이 가장 가까운 BaseLane 의 인덱스를 반환한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: Number - index

Param	Type	Description
Element	Element   String	Element 또는 ID
direction	String	

raphaelRenderer.getBoundaryOfInnerShapesGroup(Element) ⇒ [Envelope](#)

Group 의 내부 도형들의 Boundary를 반환한다.

Lane 이면 최상위 Lane의 내부 도형들의 Boundary를 반환한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: [Envelope](#) – Envelope 영역

Param	Type	Description
Element	Element   String	Element 또는 ID

raphaelRenderer.getSmallestBaseLane(Element, baseLane)

Lane 의 BaseLane 중 길이가 가장 작은 Lane 을 반환한다.

Kind: instance method of [RaphaelRenderer](#)

Param	Type	Description
Element	Element   String	Element 또는 ID
baseLane	Element	

raphaelRenderer.resizeLane(Element, offset)

Lane 을 리사이즈한다.

Kind: instance method of [RaphaelRenderer](#)

Param	Type	Description
Element	Element   String	Element 또는 ID
offset	Array.<Number>	[상, 하, 좌, 우] 각 방향으로 + 값

```
raphaelRenderer.removeLaneShape(Element)
```

Lane 을 삭제한다.

Kind: instance method of [RaphaelRenderer](#)

Param	Type	Description
Element	Element   String	Element 또는 ID

```
raphaelRenderer.getInnerShapesOfLane(Element)
```

Lane 내부 도형들을 구한다.

Kind: instance method of [RaphaelRenderer](#)

Param	Type	Description
Element	Element   String	Element 또는 ID

```
raphaelRenderer.fitLaneOrder(Element)
```

Lane 의 내부 도형들을 앞으로 이동시킨다.

Kind: instance method of [RaphaelRenderer](#)

Param	Type	Description
-------	------	-------------

`Element Element | String Element 또는 ID`

`raphaelRenderer.getRootGroupOfShape(Element) ⇒ Element`

`Shape` 가 소속된 의 최상위 그룹 앤리먼트를 반환한다.

그룹이 소속이 아닌 앤리먼트는 자신을 반환한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: Element - Element

Param            Type            Description

`Element Element | String 또는 ID`

`raphaelRenderer.checkBridgeEdge(Element)`

`Edge` 가 Gourp 사이를 넘어가는 경우 스타일에 변화를 준다.

Kind: instance method of [RaphaelRenderer](#)

Param            Type            Description

`Element Element | String 또는 ID`

`raphaelRenderer.checkAllBridgeEdge()`

모든 Edge 를 checkBridgeEdge

Kind: instance method of [RaphaelRenderer](#)

```
raphaelRenderer.getInnerShapesOfGroup(Element)
```

Group 내부의 모든 shape 을 리턴한다.

Kind: instance method of [RaphaelRenderer](#)

Param	Type	Description
Element	Element   String	Element 또는 ID

```
raphaelRenderer.getFrontForCoordinate(point) => Element
```

주어진 좌표를 포함하는 Element 중 가장 Front에 위치한 Element를 반환한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: Element - Element

Param	Type	Description
point	Array.<Number>	좌표값

```
raphaelRenderer.getFrontForBoundary(boundary) => Element
```

Boundary를 포함하는 가장 Front에 위치한 Group Element를 반환한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: Element - Element

Param	Type	Description
boundary	<a href="#">Envelope</a>	영역

```
raphaelRenderer.trimEdgeDirection(Edge, FromShape, ToShape) => Element
```

신규 Edge 의 vertices 를 연결대상 도형에 따라 설정한다

Kind: instance method of [RaphaelRenderer](#)

Returns: Element - Edge Element

Param	Type	Description
Edge	Element   String	Element 또는 ID
FromShape	Element   String	Element 또는 ID
ToShape	Element   String	Element 또는 ID

raphaelRenderer.putInnerShapeToPool(Element) ⇒ Element

Lane 또는 Pool 내부 도형들을 그룹에 포함시킨다.

Kind: instance method of [RaphaelRenderer](#)

Returns: Element - Element

Param	Type
Element	Element   String

raphaelRenderer.setDropablePool(Element) ⇒ Element

신규 Lane 또는 Pool 이 캔버스상에서 드래그하여 그려지도록 사전작업을 수행한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: Element - Element

Param	Type
Element	Element   String

```
raphaelRenderer.offDropablePool()
```

신규 Lane 또는 Pool 드랍모드를 해제한다.

Kind: instance method of [RaphaelRenderer](#)

```
raphaelRenderer.drawShape(position, shape, size, style, id) ⇒ Element
```

Shape 을 캔버스에 위치 및 사이즈 지정하여 드로잉한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [drawShape](#)

Returns: Element - Group DOM Element with geometry

Param	Type	Description
position	Array.<Number>	드로잉할 위치 좌표(중앙 기준)
shape	<a href="#">IShape</a>	Shape
size	Array.<Number>	Shape Width, Height
style	<a href="#">Style</a>   Object	스타일
id	String	Element ID 지정

Example

```
renderer.drawShape([100, 100], new OG.CircleShape(), [50, 50], {stroke: 'red'});
```

```
raphaelRenderer.drawGeom(geometry, style) ⇒ Element
```

Geometry 를 캔버스에 드로잉한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [drawGeom](#)

Returns: Element - Group DOM Element with geometry

Param	Type	Description
geometry	<a href="#">Geometry</a>	기하 객체
style	<a href="#">Style</a>   Object	스타일

```
raphaelRenderer.drawText(position, text, size, style, id) => Element
```

Text 를 캔버스에 위치 및 사이즈 지정하여 드로잉한다.

(스타일 'text-anchor': 'start' or 'middle' or 'end' 에 따라 위치 기준이 다름)

Kind: instance method of [RaphaelRenderer](#)

Overrides: [drawText](#)

Returns: Element - DOM Element

Param	Type	Description
position	Array. <Number>	드로잉할 위치 좌표(스타일 'text-anchor': 'start' or 'middle' or 'end' 에 따라 기준이 다름)
text	String	텍스트
size	Array. <Number>	Text Width, Height, Angle
style	<a href="#">Style</a>   Object	스타일
id	String	Element ID 지정

Example

```
renderer.drawText([100, 100], 'Hello', null, {'text-anchor': 'start'});
```

```
raphaelRenderer.drawImage(position, imgSrc, size, style, id) => Element
```

Image 를 캔버스에 위치 및 사이즈 지정하여 드로잉한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [drawImage](#)

Returns: Element - DOM Element

Param	Type	Description
position	Array.<Number>	드로잉할 위치 좌표(좌상단 기준)
imgSrc	String	이미지경로
size	Array.<Number>	Image Width, Height, Angle
style	<a href="#">Style</a>   Object	스타일
id	String	Element ID 지정

Example

```
renderer.drawImage([100, 100], 'img.jpg', [50, 50]);
```

```
raphaelRenderer.drawEdge(line, style, id, isSelf) ⇒ Element
```

라인을 캔버스에 드로잉한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [drawEdge](#)

Returns: Element - Group DOM Element with geometry

Param	Type	Description
line	<a href="#">Line</a>	라인
style	<a href="#">Style</a>   Object	스타일
id	String	Element ID 지정
isSelf	Boolean	셀프 연결 여부

```
raphaelRenderer.drawLabel(shapeElement, text, style) => Element
```

Shape 의 Label 을 캔버스에 위치 및 사이즈 지정하여 드로잉한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [drawLabel](#)

Returns: Element - DOM Element

Param	Type	Description
shapeElement	Element   String	Shape DOM element or ID
text	String	텍스트
style	Object	스타일

```
drawLabel~getCenterOfEdge(element) => OG.Coordinate
```

라인(꺽은선)의 중심위치를 반환한다.

Kind: inner method of [drawLabel](#)

Param	Type	Description
element	Element Edge	엘리먼트

```
raphaelRenderer.drawEdgeLabel(shapeElement, text, type) => Element
```

Edge 의 from, to Label 을 캔버스에 위치 및 사이즈 지정하여 드로잉한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [drawEdgeLabel](#)

Returns: Element - DOM Element

Param	Type	Description
shapeElement	Element   String	Shape DOM element or ID
text	String	텍스트
type	String	유형(FROM or TO)

raphaelRenderer.redrawShape(element, excludeEdgeId)

Element 에 저장된 geom, angle, image, text 정보로 shape 을 redraw 한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [redrawShape](#)

Param	Type	Description
element	Element	Shape 엘리먼트
excludeEdgeId	Array.<String>	redraw 제외할 Edge ID

raphaelRenderer.redrawConnectedEdge(element)

Shape 의 연결된 Edge 를 redraw 한다.(이동 또는 리사이즈)

Kind: instance method of [RaphaelRenderer](#)

Overrides: [redrawConnectedEdge](#)

Param	Type
element	Element

raphaelRenderer.connect(fromTerminal, toTerminal, edge, style, label, preventTrigger) ⇒ Element

두개의 터미널을 연결하고, 속성정보에 추가한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [connect](#)

Returns: Element - 연결된 Edge 엘리먼트

Param	Type	Description
fromTerminal	Element   Array.<Number>	시작점 (fromTerminal)
toTerminal	Element   Array.<Number>	끝점 (toTerminal)
edge	Element	Edge Shape
style	<a href="#">Style</a>   Object	스타일
label	String	Label
preventTrigger	Boolean	이벤트 트리거 발생 막기

```
raphaelRenderer.disconnect(element)
```

연결속성정보를 삭제한다. Edge 인 경우는 라인만 삭제하고, 일반 Shape 인 경우는 연결된 모든 Edge 를 삭제한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [disconnect](#)

Param	Type
element	Element

```
raphaelRenderer.drawDropOverGuide(element)
```

ID에 해당하는 Element 의 Edge 연결시 Drop Over 가이드를 드로잉한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [drawDropOverGuide](#)

Param	Type	Description
element	Element   String	Element 또는 ID

```
raphaelRenderer.drawGuide(element) => Object
```

ID에 해당하는 Element 의 Move & Resize 용 가이드를 드로잉한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [drawGuide](#)

Param	Type	Description
element	Element   String	Element 또는 ID

```
raphaelRenderer.removeGuide(element)
```

ID에 해당하는 Element 의 Move & Resize 용 가이드를 제거한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [removeGuide](#)

Param	Type	Description
element	Element   String	Element 또는 ID

```
raphaelRenderer.removeAllGuide()
```

모든 Move & Resize 용 가이드를 제거한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [removeAllGuide](#)

```
raphaelRenderer.drawEdgeGuide(element) => Object
```

ID에 해당하는 Edge Element 의 Move & Resize 용 가이드를 드로잉한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [drawEdgeGuide](#)

Param	Type	Description
element	Element   String	Element 또는 ID

```
raphaelRenderer.drawRubberBand(position, size, style) ⇒ Element
```

Rectangle 모양의 마우스 드래그 선택 박스 영역을 드로잉한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [drawRubberBand](#)

Returns: Element - DOM Element

Param	Type	Description
position	Array.<Number>	드로잉할 위치 좌표(좌상단)
size	Array.<Number>	Text Width, Height, Angle
style	<a href="#">Style</a>   Object	스타일

```
raphaelRenderer.removeRubberBand(root)
```

Rectangle 모양의 마우스 드래그 선택 박스 영역을 제거한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [removeRubberBand](#)

Param	Type	Description
root	Element	first, rubberBand 정보를 저장한 엘리먼트

```
raphaelRenderer.drawDraggableGuide(element) => Element
```

ID에 해당하는 Element 의 Draggable 가이드를 드로잉한다.

Kind: instance method of [RaphaelRenderer](#)

Param	Type	Description
element	Element   String	Element 또는 ID

```
raphaelRenderer.drawCollapseGuide(element) => Element
```

ID에 해당하는 Element 의 Collapse 가이드를 드로잉한다.

Kind: instance method of [RaphaelRenderer](#)

Param	Type	Description
element	Element   String	Element 또는 ID

```
raphaelRenderer.removeCollapseGuide(element)
```

ID에 해당하는 Element 의 Collapse 가이드를 제거한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [removeCollapseGuide](#)

Param	Type
element	Element

```
raphaelRenderer.group(elements) => Element
```

주어진 Shape 들을 그룹핑한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [group](#)

Returns: Element - Group Shape Element

Param	Type
elements	Array.<Element>

```
raphaelRenderer.ungroup(groupElements) => Array.<Element>
```

주어진 그룹들을 그룹해제한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [ungroup](#)

Returns: Array.<Element> - ungrouped Elements

Param	Type
groupElements	Array.<Element>

```
raphaelRenderer.addToGroup(groupElement, elements)
```

주어진 Shape 들을 그룹에 추가한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [addToGroup](#)

Param	Type
-------	------

```
groupElement Element  
elements     Array.<Element>
```

```
raphaelRenderer.collapse(element)
```

주어진 Shape 이 그룹인 경우 collapse 한다.

Kind: instance method of [RaphaelRenderer](#)

Param Type  
element Element

```
raphaelRenderer.expand(element)
```

주어진 Shape 이 그룹인 경우 expand 한다.

Kind: instance method of [RaphaelRenderer](#)

Param Type  
element Element

```
raphaelRenderer.clear()
```

드로잉된 모든 오브젝트를 클리어한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [clear](#)

```
raphaelRenderer.removeShape(element)
```

Shape 을 캔버스에서 관련된 모두를 삭제한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [removeShape](#)

Param	Type	Description
element	Element   String	Element 또는 ID

```
raphaelRenderer.remove(element)
```

ID에 해당하는 Element 를 캔버스에서 제거한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [remove](#)

Param	Type	Description
element	Element   String	Element 또는 ID

```
raphaelRenderer.removeChild(element)
```

하위 엘리먼트만 제거한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [removeChild](#)

Param	Type	Description
element	Element   String	Element 또는 ID

```
raphaelRenderer.getRootElement() => Element
```

랜더러 캔버스 Root Element 를 반환한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [getRootElement](#)

Returns: Element - Element

```
raphaelRenderer.getRootGroup() => Element
```

랜더러 캔버스 Root Group Element 를 반환한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: Element - Element

```
raphaelRenderer.getElementByPoint(position) => Element
```

주어진 지점을 포함하는 Top Element 를 반환한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [getElementByPoint](#)

Returns: Element - Element

Param	Type	Description
positionArray.<Number>	위치 좌표	

```
raphaelRenderer.getElementsByBBox(envelope) => Array.<Element>
```

주어진 Boundary Box 영역에 포함되는 Shape(GEOM, TEXT, IMAGE) Element 를 반환한다.

모든 vertices를 포함한 엘리먼트를 반환한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: Array.<Element> - Element

Param	Type	Description
envelope	<a href="#">Envelope</a>	Boundary Box 영역

```
raphaelRenderer.setAttr(element, attribute)
```

엘리먼트에 속성값을 설정한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [setAttr](#)

Param	Type	Description
element	Element   String	Element 또는 ID
attributeObject		속성값

```
raphaelRenderer.getAttr(element, attrName) => Object
```

엘리먼트 속성값을 반환한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [getAttr](#)

Returns: Object - attribute 속성값

Param	Type	Description
element	Element   String	Element 또는 ID
attrName	String	속성이름

```
raphaelRenderer.setShapeStyle(element, style)
```

Shape 의 스타일을 변경한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [setShapeStyle](#)

Param	Type	Description
element	Element   String	Element 또는 ID
style	Object	스타일

```
raphaelRenderer.toFront(element)
```

ID에 해당하는 Element 를 최상단 레이어로 이동한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [toFront](#)

Param	Type	Description
element	Element   String	Element 또는 ID

```
raphaelRenderer.toBack(element)
```

ID에 해당하는 Element 를 최하단 레이어로 이동한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [toBack](#)

Param	Type	Description
element	Element   String	Element 또는 ID

```
raphaelRenderer.bringForward(element)
```

ID에 해당하는 Element 를 앞으로 한단계 이동한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [bringForward](#)

Param	Type	Description
element	Element   String	Element 또는 ID

```
raphaelRenderer.sendBackward(element)
```

ID에 해당하는 Element 를 뒤로 한단계 이동한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [sendBackward](#)

Param	Type	Description
element	Element   String	Element 또는 ID

```
raphaelRenderer.getCanvasSize() => Array.<Number>
```

랜더러 캔버스의 사이즈(Width, Height)를 반환한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [getCanvasSize](#)

Returns: Array.<Number> - Canvas Width, Height

```
raphaelRenderer.setCanvasSize(size)
```

랜더러 캔버스의 사이즈(Width, Height)를 변경한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [setCanvasSize](#)

Param	Type	Description
size	Array.<Number>	Canvas Width, Height

```
raphaelRenderer.fitCanvasSize(minSize, fitScale)
```

랜더러 캔버스의 사이즈(Width, Height)를 실제 존재하는 Shape 의 영역에 맞게 변경한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [fitCanvasSize](#)

Param	Type	Description
minSize	Array.<Number>	Canvas 최소 Width, Height
fitScale	Boolean	주어진 minSize 에 맞게 fit 여부(Default:false)

```
raphaelRenderer.setViewBox(position, size, isFit)
```

새로운 View Box 영역을 설정한다. (ZoomIn & ZoomOut 가능)

Kind: instance method of [RaphaelRenderer](#)

Overrides: [setViewBox](#)

Param	Type	Description
position	Array.<Number>	위치 좌표(좌상단 기준)
size	Array.<Number>	Canvas Width, Height
isFit	Boolean	Fit 여부

raphaelRenderer.getScale() ⇒ Number

Scale 을 반환한다. (리얼 사이즈 : Scale = 1)

Kind: instance method of [RaphaelRenderer](#)

Overrides: [getScale](#)

Returns: Number - 스케일값

raphaelRenderer.setScale(scale)

Scale 을 설정한다. (리얼 사이즈 : Scale = 1)

Kind: instance method of [RaphaelRenderer](#)

Overrides: [setScale](#)

Param	Type	Description
scale	Number	스케일값

raphaelRenderer.show(element)

ID에 해당하는 Element 를 캔버스에서 show 한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [show](#)

Param	Type	Description
element	Element   String	Element 또는 ID

```
raphaelRenderer.hide(element)
```

ID에 해당하는 Element 를 캔버스에서 hide 한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [hide](#)

Param	Type	Description
element	Element   String	Element 또는 ID

```
raphaelRenderer.appendChild(srcElement, targetElement) ⇒ Element
```

Source Element 를 Target Element 아래에 append 한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [appendChild](#)

Returns: Element - Source Element

Param	Type	Description
srcElement	Element   String	Element 또는 ID
targetElement	Element   String	Element 또는 ID

```
raphaelRenderer.insertAfter(srcElement, targetElement) ⇒ Element
```

Source Element 를 Target Element 이후에 insert 한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [insertAfter](#)

Returns: Element - Source Element

Param	Type	Description
srcElement	Element   String	Element 또는 ID
targetElement	Element   String	Element 또는 ID

```
raphaelRenderer.insertBefore(srcElement, targetElement) => Element
```

Source Element 를 Target Element 이전에 insert 한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [insertBefore](#)

Returns: Element - Source Element

Param	Type	Description
srcElement	Element   String	Element 또는 ID
targetElement	Element   String	Element 또는 ID

```
raphaelRenderer.move(element, offset) => Element
```

해당 Element 를 가로, 세로 Offset 만큼 이동한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [move](#)

Returns: Element - Element

Param	Type	Description
element	Element   String	Element 또는 ID
offset	Array.<Number>	[가로, 세로]

```
raphaelRenderer.moveCentroid(element, position) => Element
```

주어진 중심좌표로 해당 Element 를 이동한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [moveCentroid](#)

Returns: Element - Element

Param	Type	Description
element	Element   String	Element 또는 ID
position	Array.<Number>	[x, y]

```
raphaelRenderer.rotate(element, angle) => Element
```

중심 좌표를 기준으로 주어진 각도 만큼 회전한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [rotate](#)

Returns: Element - Element

Param	Type	Description
element	Element   String	Element 또는 ID
angle	Number	각도

```
raphaelRenderer.resize(element, offset) => Element
```

상, 하, 좌, 우 외곽선을 이동한 만큼 리사이즈 한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [resize](#)

Returns: Element - Element

Param	Type	Description
element	Element   String	Element 또는 ID
offset	Array.<Number>	[상, 하, 좌, 우] 각 방향으로 + 값

```
raphaelRenderer.resizeBox(element, size) => Element
```

중심좌표는 고정한 채 Bounding Box 의 width, height 를 리사이즈 한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [resizeBox](#)

Returns: Element - Element

Param	Type	Description
element	Element   String	Element 또는 ID
size	Array.<Number>	[Width, Height]

```
raphaelRenderer.clone(element) => Element
```

노드 Element 를 복사한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [clone](#)

Returns: Element - Element

Param	Type	Description
element	Element   String	Element 또는 ID

```
raphaelRenderer.getElementById(id) => Element
```

ID로 Node Element 를 반환한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [getElementById](#)

Returns: Element - Element

Param	Type
id	String

```
raphaelRenderer.getElementsByType(shapeType, excludeType) => Array.<Element>
```

Shape 타입에 해당하는 Node Element 들을 반환한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: Array.<Element> - Element's Array

Param	Type	Description
shapeType	String	Shape 타입(GEOM, HTML, IMAGE, EDGE, GROUP), Null 이면 모든 타입
excludeType	String	제외 할 타입

```
raphaelRenderer.getBBox(element) => Object
```

해당 엘리먼트의 BoundingBox 영역 정보를 반환한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [getBBox](#)

Returns: Object - {width, height, x, y, x2, y2}

Param	Type
element	Element   String

```
raphaelRenderer.getRootBBox() => Object
```

부모노드기준으로 캔버스 루트 엘리먼트의 BoundingBox 영역 정보를 반환한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [getRootBBox](#)

Returns: Object - {width, height, x, y, x2, y2}

```
raphaelRenderer.getRealRootBBox() => Object
```

부모노드기준으로 캔버스 루트 엘리먼트의 실제 Shape 이 차지하는 BoundingBox 영역 정보를 반환한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: Object - {width, height, x, y, x2, y2}

```
raphaelRenderer.getContainer() => Element
```

캔버스의 컨테이너 DOM element 를 반환한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [getContainer](#)

Returns: Element - 컨테이너

raphaelRenderer.isSVG() => Boolean

SVG 인지 여부를 반환한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [isSVG](#)

Returns: Boolean - svg 여부

raphaelRenderer.isVML() => Boolean

VML 인지 여부를 반환한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [isVML](#)

Returns: Boolean - vml 여부

raphaelRenderer.getPrevEdges(element) => Array.<Element>

연결된 이전 Edge Element 들을 반환한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: Array.<Element> - Previous Element's Array

Param	Type	Description
element	Element   String	Element 또는 ID

```
raphaelRenderer.getNextEdges(element) ⇒ Array.<Element>
```

연결된 이후 Edge Element 들을 반환한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: Array.<Element> - Previous Element's Array

Param	Type	Description
element	Element   String	Element 또는 ID

```
raphaelRenderer.getPrevShapes(element) ⇒ Array.<Element>
```

연결된 이전 노드 Element 들을 반환한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: Array.<Element> - Previous Element's Array

Param	Type	Description
element	Element   String	Element 또는 ID

```
raphaelRenderer.getPrevShapeIds(element) ⇒ Array.<String>
```

연결된 이전 노드 Element ID들을 반환한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: Array.<String> - Previous Element Id's Array

Param	Type	Description
element	Element   String	Element 또는 ID

```
raphaelRenderer.getNextShapes(element) ⇒ Array.<Element>
```

연결된 이후 노드 Element 들을 반환한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: Array.<Element> - Previous Element's Array

Param	Type	Description
element	Element   String	Element 또는 ID

```
raphaelRenderer.getNextShapeIds(element) ⇒ Array.<String>
```

연결된 이후 노드 Element ID들을 반환한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: Array.<String> - Previous Element Id's Array

Param	Type	Description
element	Element   String	Element 또는 ID

```
raphaelRenderer.isTopGroup(Element) ⇒ boolean
```

최상위 그룹 엘리먼트인지 반환한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: boolean - true false

Param	Type	Description
-------	------	-------------

Element Element 엘리먼트

raphaelRenderer.getParent(element) ⇒ Element

부모 엘리먼트를 반환한다. 부모가 루트일때는 반환하지 않는다.

Kind: instance method of [RaphaelRenderer](#)

Returns: Element - element 엘리먼트

Param	Type	Description
element	Element	엘리먼트

raphaelRenderer.getChildren(element) ⇒ Array

그룹의 하위 엘리먼트를 반환한다.

Kind: instance method of [RaphaelRenderer](#)

Returns: Array - Elements

Param	Type	Description
element	Element	엘리먼트

raphaelRenderer.isGroup(element) ⇒ boolean

그룹의 Shape 인지 반환한다. RootGroup 일 경우는 제외.

Kind: instance method of [RaphaelRenderer](#)

Returns: boolean - true false

Param	Type	Description
element	Element	엘리먼트

```
raphaelRenderer.getAllShapes() => Array
```

캔버스의 모든 Shape 들을 리턴

Kind: instance method of [RaphaelRenderer](#)

Returns: Array - Elements

```
raphaelRenderer.getAllEdges() => Array
```

캔버스의 모든 Edge를 리턴

Kind: instance method of [RaphaelRenderer](#)

Returns: Array - Edge Elements

```
raphaelRenderer.getAllNotEdges() => Array
```

캔버스의 모든 Edge 가 아닌 shpaes 를 리턴

Kind: instance method of [RaphaelRenderer](#)

Returns: Array - Edge Elements

```
raphaelRenderer.isEdge() => boolean
```

Edge 여부를 판단.

Kind: instance method of [RaphaelRenderer](#)

Returns: boolean - true false

raphaelRenderer.isShape() ⇒ boolean

Shape 여부를 판단.

Kind: instance method of [RaphaelRenderer](#)

Returns: boolean - true false

raphaelRenderer.initHistory()

캔버스의 히스토리를 초기화한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [initHistory](#)

raphaelRenderer.addHistory()

캔버스에 히스토리를 추가한다.

Kind: instance method of [RaphaelRenderer](#)

Overrides: [addHistory](#)

raphaelRenderer.undo()

캔버스의 Undo

Kind: instance method of [RaphaelRenderer](#)

Overrides: [undo](#)

raphaelRenderer.redo()

캔버스의 Redo

Kind: instance method of [RaphaelRenderer](#)

Overrides: [redo](#)

OG.marker : object

Kind: static namespace of [OG](#)

- [.marker](#) : object
  - [.ArrowMarker](#) ⇐ [IMarker](#)
    - [new OG.marker.ArrowMarker\(\)](#)
    - [.MARKER\\_ID](#) : String
    - [.geom](#) : [Geometry](#)
    - [.createMarker\(\)](#) ⇒ \*
  - [.CircleMarker](#) ⇐ [IMarker](#)

- `new OG.marker.CircleMarker()`
- `.MARKER_ID` : String
- `.geom` : Geometry
- `.createMarker()` ⇒ \*

- [IMarker](#)

- `new OG.marker.IMarker()`
- `.MARKER_ID` : String
- `.geom` : Geometry
- `.createMarker()` ⇒ \*

- [RectangleMarker](#) ⇐ IMarker

- `new OG.marker.RectangleMarker()`
- `.MARKER_ID` : String
- `.geom` : Geometry
- `.createMarker()` ⇒ \*

- [SwitchLMarker](#) ⇐ IMarker

- `new OG.marker.SwitchLMarker()`
- `.MARKER_ID` : String
- `.geom` : Geometry
- `.createMarker()` ⇒ \*

- [SwitchRMarker](#) ⇐ IMarker

- `new OG.marker.SwitchRMarker()`

- [.MARKER\\_ID](#) : String
  - [.geom](#) : [Geometry](#)
  - [.createMarker\(\)](#) ⇒ \*
- 
- [.SwitchXMarker](#) ⇐ [IMarker](#)
    - [new OG.marker.SwitchXMarker\(\)](#)
    - [.MARKER\\_ID](#) : String
    - [.geom](#) : [Geometry](#)
    - [.createMarker\(\)](#) ⇒ \*

marker.ArrowMarker ⇐ [IMarker](#)

Kind: static class of [marker](#)

Extends: [IMarker](#)

Requires: module:OG.common.\* , module:OG.geometry.\*

Author: [Seungpil Park \(mailto:sppark@uengine.org\)](mailto:Seungpil.Park@sppark@uengine.org)

- [.ArrowMarker](#) ⇐ [IMarker](#)
  - [new OG.marker.ArrowMarker\(\)](#)
  - [.MARKER\\_ID](#) : String
  - [.geom](#) : [Geometry](#)
  - [.createMarker\(\)](#) ⇒ \*

```
new OG.marker.ArrowMarker()
```

Rectangle Maker

```
arrowMarker.MARKER_ID : String
```

marker 을 구분하는 marker ID(marker 클래스명과 일치)

Kind: instance property of [ArrowMarker](#)

Overrides: [MARKER\\_ID](#)

```
arrowMarker.geom : Geometry
```

marker 모양을 나타내는 공간기하 객체(Geometry)

Kind: instance property of [ArrowMarker](#)

Overrides: [geom](#)

```
arrowMarker.createMarker() => *
```

드로잉할 marker 를 생성하여 반환한다.

Kind: instance method of [ArrowMarker](#)

Overrides: [createMarker](#)

Returns: \* - Marker 정보

marker.CircleMarker <= [IMarker](#)

Kind: static class of [marker](#)

Extends: [IMarker](#)

Requires: module:OG.common.\* , module:OG.geometry.\*

Author: [Seungpil Park \(mailto:sppark@uengine.org\)](mailto:Seungpil.Park@sppark@uengine.org)

- [.CircleMarker](#) <= [IMarker](#)

- [new OG.marker.CircleMarker\(\)](#)
- [.MARKER\\_ID](#) : String
- [.geom](#) : [Geometry](#)
- [.createMarker\(\)](#) ⇒ \*

[new OG.marker.CircleMarker\(\)](#)

Circle Marker

circleMarker.MARKER\_ID : String

marker 을 구분하는 marker ID(marker 클래스명과 일치)

Kind: instance property of [CircleMarker](#)

Overrides: [MARKER\\_ID](#)

circleMarker.geom : [Geometry](#)

marker 모양을 나타내는 공간기하 객체(Geometry)

Kind: instance property of [CircleMarker](#)

Overrides: [geom](#)

circleMarker.createMarker() ⇒ \*

드로잉할 marker 를 생성하여 반환한다.

Kind: instance method of [CircleMarker](#)

Overrides: [createMarker](#)

Returns: \* - Marker 정보

marker.IMarker

Kind: static class of [marker](#)

Requires: module:OG.common.\* , module:OG.geometry.\*

Author: [Seungpil Park \(mailto:spark@uengine.org\)](mailto:Seungpil.Park@uengine.org)

- [.IMarker](#)

- [new OG.marker.IMarker\(\)](#)

- [.MARKER\\_ID](#) : String
- [.geom](#) : [Geometry](#)
- [.createMarker\(\)](#) ⇒ \*

new OG.marker.IMarker()

도형 Path 의 Marker 정보 최상위 인터페이스

iMarker.MARKER\_ID : String

marker 을 구분하는 marker ID(marker 클래스명과 일치)

Kind: instance property of [IMarker](#)

iMarker.geom : [Geometry](#)

marker 모양을 나타내는 공간기하 객체(Geometry)

Kind: instance property of [IMarker](#)

iMarker.createMarker() ⇒ \*

드로잉할 marker 를 생성하여 반환한다.

Kind: instance abstract method of [IMarker](#)

Returns: \* - Marker 정보

marker.RectangleMarker ⇐ [IMarker](#)

Kind: static class of [marker](#)

Extends: [IMarker](#)

Requires: module:OG.common.\* , module:OG.geometry.\*

Author: [Seungpil Park \(mailto:spark@uengine.org\)](mailto:Seungpil_Park@uengine.org)

- [.RectangleMarker](#) ⇐ [IMarker](#)

- [new OG.marker.RectangleMarker\(\)](#)
- [.MARKER\\_ID](#) : String
- [.geom](#) : [Geometry](#)
- [.createMarker\(\)](#) ⇒ \*

`new OG.marker.RectangleMarker()`

Rectangle Maker

`rectangleMarker.MARKER_ID` : String

marker 을 구분하는 marker ID(marker 클래스명과 일치)

Kind: instance property of [RectangleMarker](#)

Overrides: [MARKER\\_ID](#)

rectangleMarker.geom : [Geometry](#)

marker 모양을 나타내는 공간기하 객체(Geometry)

Kind: instance property of [RectangleMarker](#)

Overrides: [geom](#)

rectangleMarker.createMarker() => \*

드로잉할 marker 를 생성하여 반환한다.

Kind: instance method of [RectangleMarker](#)

Overrides: [createMarker](#)

Returns: \* - Marker 정보

marker.SwitchLMarker <= [IMarker](#)

Kind: static class of [marker](#)

Extends: [IMarker](#)

Requires: module:OG.common.\* , module:OG.geometry.\*

Author: [Seungpil Park \(mailto:spspark@uengine.org\)](mailto:Seungpil.Park@uengine.org)

- [.SwitchLMarker](#) ⇐ [IMarker](#)
  - [new OG.marker.SwitchLMarker\(\)](#)
  - [.MARKER\\_ID](#) : String
  - [.geom](#) : [Geometry](#)
  - [.createMarker\(\)](#) ⇒ \*

`new OG.marker.SwitchLMarker()`

Rectangle Maker

`switchLMarker.MARKER_ID` : String

marker 을 구분하는 marker ID(marker 클래스명과 일치)

Kind: instance property of [SwitchLMarker](#)

Overrides: [MARKER\\_ID](#)

`switchLMarker.geom` : [Geometry](#)

marker 모양을 나타내는 공간기하 객체(Geometry)

Kind: instance property of [SwitchLMarker](#)

Overrides: [geom](#)

```
switchLMarker.createMarker() => *
```

드로잉할 marker 를 생성하여 반환한다.

Kind: instance method of [SwitchLMarker](#)

Overrides: [createMarker](#)

Returns: \* - Marker 정보

marker.SwitchRMarker <= [IMarker](#)

Kind: static class of [marker](#)

Extends: [IMarker](#)

Requires: module:OG.common.\* , module:OG.geometry.\*

Author: [Seungpil Park \(mailto:sppark@uengine.org\)](mailto:Seungpil_Park (mailto:sppark@uengine.org))

- [.SwitchRMarker](#) <= [IMarker](#)

- [new OG.marker.SwitchRMarker\(\)](#)
- [.MARKER\\_ID](#) : String
- [.geom](#) : [Geometry](#)
- [.createMarker\(\)](#) => \*

```
new OG.marker.SwitchRMarker()
```

Rectangle Maker

switchRMarker.MARKER\_ID : String

marker 을 구분하는 marker ID(marker 클래스명과 일치)

Kind: instance property of [SwitchRMarker](#)

Overrides: [MARKER\\_ID](#)

switchRMarker.geom : [Geometry](#)

marker 모양을 나타내는 공간기하 객체(Geometry)

Kind: instance property of [SwitchRMarker](#)

Overrides: [geom](#)

switchRMarker.createMarker() ⇒ \*

드로잉할 marker 를 생성하여 반환한다.

Kind: instance method of [SwitchRMarker](#)

Overrides: [createMarker](#)

Returns: \* - Marker 정보

marker.SwitchXMarker ⇐ [IMarker](#)

Kind: static class of [marker](#)

Extends: [IMarker](#)

Requires: module:OG.common.\* , module:OG.geometry.\*

Author: [Seungpil Park \(mailto:sppark@uengine.org\)](mailto:Seungpil.Park@sppark@uengine.org)

- [.SwitchXMarker](#) ⇐ [IMarker](#)

- [new OG.marker.SwitchXMarker\(\)](#)
- [.MARKER\\_ID](#) : String
- [.geom](#) : [Geometry](#)
- [.createMarker\(\)](#) ⇒ \*

`new OG.marker.SwitchXMarker()`

Rectangle Maker

`switchXMarker.MARKER_ID` : String

marker 을 구분하는 marker ID(marker 클래스명과 일치)

Kind: instance property of [SwitchXMarker](#)

Overrides: [MARKER\\_ID](#)

`switchXMarker.geom` : [Geometry](#)

marker 모양을 나타내는 공간기하 객체(Geometry)

Kind: instance property of [SwitchXMarker](#)

Overrides: [geom](#)

switchXMarker.createMarker() ⇒ \*

드로잉할 marker 를 생성하여 반환한다.

Kind: instance method of [SwitchXMarker](#)

Overrides: [createMarker](#)

Returns: \* - Marker 정보

OG.pattern : object

Kind: static namespace of [OG](#)

- [.pattern](#) : object
  - [.HatchedPattern](#) ⇐ [IPattern](#)
    - [new OG.pattern.HatchedPattern\(\)](#)
    - [.PATTERN\\_ID](#) : String
    - [.geom](#) : [Geometry](#)
    - [.createPattern\(\)](#) ⇒ \*
  - [.IPattern](#)

- [new OG.pattern.IPattern\(\)](#)
  - [.PATTERN\\_ID](#) : String
  - [.geom](#) : [Geometry](#)
  - [.createPattern\(\)](#) ⇒ \*
- [.RectPattern](#) ⇐ [IPattern](#)
    - [new OG.pattern.RectPattern\(\)](#)
    - [.PATTERN\\_ID](#) : String
    - [.geom](#) : [Geometry](#)
    - [.createPattern\(\)](#) ⇒ \*

pattern.HatchedPattern ⇐ [IPattern](#)

Kind: static class of [pattern](#)

Extends: [IPattern](#)

Requires: module:OG.common.\* , module:OG.geometry.\*

Author: [Seungpil Park \(mailto:sppark@uengine.org\)](mailto:Seungpil.Park@sppark@uengine.org)

- [.HatchedPattern](#) ⇐ [IPattern](#)
  - [new OG.pattern.HatchedPattern\(\)](#)
  - [.PATTERN\\_ID](#) : String
  - [.geom](#) : [Geometry](#)

- [\\_.createPattern\(\)](#) ⇒ \*

new OG.pattern.HatchedPattern()

### Hatched Pattern

hatchedPattern.PATTERN\_ID : String

pattern 을 구분하는 pattern ID(pattern 클래스명과 일치)

Kind: instance property of [HatchedPattern](#)

Overrides: [PATTERN\\_ID](#)

hatchedPattern.geom : [Geometry](#)

pattern 모양을 나타내는 공간기하 객체(Geometry)

Kind: instance property of [HatchedPattern](#)

Overrides: [geom](#)

hatchedPattern.createPattern() ⇒ \*

드로잉할 pattern 를 생성하여 반환한다.

Kind: instance method of [HatchedPattern](#)

Overrides: [createPattern](#)

Returns: \* - pattern 정보

pattern.IPattern

Kind: static class of [pattern](#)

Requires: module:OG.common.\* , module:OG.geometry.\*

Author: [Seungpil Park \(mailto:spark@uengine.org\)](mailto:Seungpil.Park@uengine.org)

- [IPattern](#)
  - [new OG.pattern.IPattern\(\)](#)
  - [.PATTERN\\_ID](#) : String
  - [.geom](#) : Geometry
  - [.createPattern\(\)](#) ⇒ \*

new OG.pattern.IPattern()

도형 Pattern 정보 최상위 인터페이스

iPattern.PATTERN\_ID : String

pattern 을 구분하는 pattern ID(pattern 클래스명과 일치)

Kind: instance property of [IPattern](#)

`iPattern.geom : Geometry`

`pattern` 모양을 나타내는 공간기하 객체(Geometry)

Kind: instance property of [IPattern](#)

`iPattern.createPattern() ⇒ *`

드로잉할 `pattern` 를 생성하여 반환한다.

Kind: instance abstract method of [IPattern](#)

Returns: \* - `pattern` 정보

`pattern.RectPattern ⇐ IPattern`

Kind: static class of [pattern](#)

Extends: [IPattern](#)

Requires: module:OG.common.\* , module:OG.geometry.\*

Author: [Seungpil Park \(mailto:spspark@uengine.org\)](mailto:Seungpil.Park@uengine.org)

- [.RectPattern ⇐ IPattern](#)

- [new OG.pattern.RectPattern\(\)](#)
  - [.PATTERN\\_ID : String](#)

- [.geom](#) : [Geometry](#)
- [.createPattern\(\)](#) ⇒ \*

new OG.pattern.RectPattern()

Rect Pattern

rectPattern.PATTERN\_ID : String

pattern 을 구분하는 pattern ID(pattern 클래스명과 일치)

Kind: instance property of [RectPattern](#)

Overrides: [PATTERN\\_ID](#)

rectPattern.geom : [Geometry](#)

pattern 모양을 나타내는 공간기하 객체(Geometry)

Kind: instance property of [RectPattern](#)

Overrides: [geom](#)

rectPattern.createPattern() ⇒ \*

드로잉할 pattern 를 생성하여 반환한다.

Kind: instance method of [RectPattern](#)

Overrides: [createPattern](#)

Returns: \* - pattern 정보

OG.shape : object

Kind: static namespace of [OG](#)

- [.shape](#) : object
  - [.CircleShape](#) ⇐ [GeomShape](#)
    - [new OG.shape.CircleShape\(label\)](#)
    - [.TYPE](#) : String
    - [.SHAPE\\_ID](#) : String
    - [.geom](#) : [Geometry](#)
    - [.label](#) : String
    - [.isCollapsed](#) : Boolean
    - [.SELECTABLE](#) : Boolean
    - [.MOVABLE](#) : Boolean
    - [.RESIZABLE](#) : Boolean
    - [.CONNECTABLE](#) : Boolean
    - [.ENABLE\\_FROM](#) : Boolean
    - [.ENABLE\\_TO](#) : Boolean
    - [.SELF\\_CONNECTABLE](#) : Boolean
    - [.CONNECT\\_CLONEABLE](#) : Boolean

- .CONNECT\_REQUIRED : Boolean
- .CONNECT\_STYLE\_CHANGE : Boolean
- .DELETABLE : Boolean
- .LABEL\_EDITABLE : Boolean
- .data : Object
- .textList : Array
- .clone() ⇒ IShape
- .createShape() ⇒ \*

- .EdgeShape ⇐ IShape

- new OG.shape.EdgeShape(from, to, label, fromLabel, toLabel)
- .from : Array.<Number>
- .to : Array.<Number>
- .fromLabel : String
- .toLabel : String
- .TYPE : String
- .SHAPE\_ID : String
- .geom : Geometry
- .label : String
- .isCollapsed : Boolean
- .SELECTABLE : Boolean
- .MOVABLE : Boolean
- .RESIZABLE : Boolean
- .CONNECTABLE : Boolean
- .ENABLE\_FROM : Boolean
- .ENABLE\_TO : Boolean

- .SELF\_CONNECTABLE : Boolean
- .CONNECT\_CLONEABLE : Boolean
- .CONNECT\_REQUIRED : Boolean
- .CONNECT\_STYLE\_CHANGE : Boolean
- .DELETABLE : Boolean
- .LABEL\_EDITABLE : Boolean
- .data : Object
- .textList : Array
- .createShape() ⇒ \*
- .clone() ⇒ IShape

- .EllipseShape ⇐ GeomShape

- new OG.shape.EllipseShape(label)
- .TYPE : String
- .SHAPE\_ID : String
- .geom : Geometry
- .label : String
- .isCollapsed : Boolean
- .SELECTABLE : Boolean
- .MOVABLE : Boolean
- .RESIZABLE : Boolean
- .CONNECTABLE : Boolean
- .ENABLE\_FROM : Boolean
- .ENABLE\_TO : Boolean
- .SELF\_CONNECTABLE : Boolean

- [.CONNECT\\_CLONEABLE](#) : Boolean
- [.CONNECT\\_REQUIRED](#) : Boolean
- [.CONNECT\\_STYLE\\_CHANGE](#) : Boolean
- [.DELETABLE](#) : Boolean
- [.LABEL\\_EDITABLE](#) : Boolean
- [.data](#) : Object
- [.textList](#) : Array
- [.clone\(\)](#) ⇒ [IShape](#)
- [.createShape\(\)](#) ⇒ \*

- [.GeomShape](#) ⇐ [IShape](#)

- [new OG.shape.GemShape\(\)](#)
- [.TYPE](#) : String
- [.SHAPE\\_ID](#) : String
- [.geom](#) : [Geometry](#)
- [.label](#) : String
- [.isCollapsed](#) : Boolean
- [.SELECTABLE](#) : Boolean
- [.MOVABLE](#) : Boolean
- [.RESIZABLE](#) : Boolean
- [.CONNECTABLE](#) : Boolean
- [.ENABLE\\_FROM](#) : Boolean
- [.ENABLE\\_TO](#) : Boolean
- [.SELF\\_CONNECTABLE](#) : Boolean
- [.CONNECT\\_CLONEABLE](#) : Boolean
- [.CONNECT\\_REQUIRED](#) : Boolean

- .CONNECT\_STYLE\_CHANGE : Boolean
- .DELETABLE : Boolean
- .LABEL\_EDITABLE : Boolean
- .data : Object
- .textList : Array
- .createShape() ⇒ \*
- .clone() ⇒ IShape
- .GroupShape ⇐ IShape
  - new OG.shape.GroupShape(label)
  - .GROUP\_DROPABLE : Boolean
  - .GROUP\_COLLAPSIBLE : Boolean
  - .TYPE : String
  - .SHAPE\_ID : String
  - .geom : Geometry
  - .label : String
  - .isCollapsed : Boolean
  - .SELECTABLE : Boolean
  - .MOVABLE : Boolean
  - .RESIZABLE : Boolean
  - .CONNECTABLE : Boolean
  - .ENABLE\_FROM : Boolean
  - .ENABLE\_TO : Boolean
  - .SELF\_CONNECTABLE : Boolean
  - .CONNECT\_CLONEABLE : Boolean

- .CONNECT\_REQUIRED : Boolean
  - .CONNECT\_STYLE\_CHANGE : Boolean
  - .DELETABLE : Boolean
  - .LABEL\_EDITABLE : Boolean
  - .data : Object
  - .textList : Array
  - .createShape() ⇒ \*
  - .clone() ⇒ IShape
- 
- .HorizontalLaneShape ⇐ GroupShape
    - new OG.shape.HorizontalLaneShape(label)
    - .GROUP\_DROPABLE : Boolean
    - .GROUP\_COLLAPSIBLE : Boolean
    - .TYPE : String
    - .SHAPE\_ID : String
    - .geom : Geometry
    - .label : String
    - .isCollapsed : Boolean
    - .SELECTABLE : Boolean
    - .MOVABLE : Boolean
    - .RESIZABLE : Boolean
    - .CONNECTABLE : Boolean
    - .ENABLE\_FROM : Boolean
    - .ENABLE\_TO : Boolean
    - .SELF\_CONNECTABLE : Boolean
    - .CONNECT\_CLONEABLE : Boolean

- .CONNECT\_REQUIRED : Boolean
  - .CONNECT\_STYLE\_CHANGE : Boolean
  - .DELETABLE : Boolean
  - .LABEL\_EDITABLE : Boolean
  - .data : Object
  - .textList : Array
  - .createShape() ⇒ \*
  - .clone() ⇒ IShape
- 
- .HorizontalPoolShape ⇐ GroupShape
    - new OG.shape.HorizontalPoolShape(label)
    - .GROUP\_DROPABLE : Boolean
    - .GROUP\_COLLAPSIBLE : Boolean
    - .TYPE : String
    - .SHAPE\_ID : String
    - .geom : Geometry
    - .label : String
    - .isCollapsed : Boolean
    - .SELECTABLE : Boolean
    - .MOVABLE : Boolean
    - .RESIZABLE : Boolean
    - .CONNECTABLE : Boolean
    - .ENABLE\_FROM : Boolean
    - .ENABLE\_TO : Boolean
    - .SELF\_CONNECTABLE : Boolean

- .CONNECT\_CLONEABLE : Boolean
- .CONNECT\_REQUIRED : Boolean
- .CONNECT\_STYLE\_CHANGE : Boolean
- .DELETABLE : Boolean
- .LABEL\_EDITABLE : Boolean
- .data : Object
- .textList : Array
- .createShape() ⇒ \*
- .clone() ⇒ IShape

- .HtmlShape ⇐ IShape

- new OG.shape.HtmlShape(html, label)
- .html : String
- .angle : Number
- .TYPE : String
- .SHAPE\_ID : String
- .geom : Geometry
- .label : String
- .isCollapsed : Boolean
- .SELECTABLE : Boolean
- .MOVABLE : Boolean
- .RESIZABLE : Boolean
- .CONNECTABLE : Boolean
- .ENABLE\_FROM : Boolean
- .ENABLE\_TO : Boolean
- .SELF\_CONNECTABLE : Boolean

- .CONNECT\_CLONEABLE : Boolean
- .CONNECT\_REQUIRED : Boolean
- .CONNECT\_STYLE\_CHANGE : Boolean
- .DELETABLE : Boolean
- .LABEL\_EDITABLE : Boolean
- .data : Object
- .textList : Array
- .createShape() ⇒ \*
- .clone() ⇒ IShape

- .IShape

- new OG.shape.IShape()
- .TYPE : String
- .SHAPE\_ID : String
- .geom : Geometry
- .label : String
- .isCollapsed : Boolean
- .SELECTABLE : Boolean
- .MOVABLE : Boolean
- .RESIZABLE : Boolean
- .CONNECTABLE : Boolean
- .ENABLE\_FROM : Boolean
- .ENABLE\_TO : Boolean
- .SELF\_CONNECTABLE : Boolean
- .CONNECT\_CLONEABLE : Boolean

- .CONNECT\_REQUIRED : Boolean
- .CONNECT\_STYLE\_CHANGE : Boolean
- .DELETABLE : Boolean
- .LABEL\_EDITABLE : Boolean
- .data : Object
- .textList : Array
- .createShape() ⇒ \*
- .clone() ⇒ IShape

- .ImageShape ⇐ IShape

- new OG.shape.ImageShape(image, label)
- .image : String
- .angle : Number
- .TYPE : String
- .SHAPE\_ID : String
- .geom : Geometry
- .label : String
- .isCollapsed : Boolean
- .SELECTABLE : Boolean
- .MOVABLE : Boolean
- .RESIZABLE : Boolean
- .CONNECTABLE : Boolean
- .ENABLE\_FROM : Boolean
- .ENABLE\_TO : Boolean
- .SELF\_CONNECTABLE : Boolean
- .CONNECT\_CLONEABLE : Boolean

- .CONNECT\_REQUIRED : Boolean
  - .CONNECT\_STYLE\_CHANGE : Boolean
  - .DELETABLE : Boolean
  - .LABEL\_EDITABLE : Boolean
  - .data : Object
  - .textList : Array
  - .createShape() ⇒ \*
  - .clone() ⇒ IShape
- 
- .RectangleShape ⇐ GeomShape
    - new OG.shape.RectangleShape(label)
    - .TYPE : String
    - .SHAPE\_ID : String
    - .geom : Geometry
    - .label : String
    - .isCollapsed : Boolean
    - .SELECTABLE : Boolean
    - .MOVABLE : Boolean
    - .RESIZABLE : Boolean
    - .CONNECTABLE : Boolean
    - .ENABLE\_FROM : Boolean
    - .ENABLE\_TO : Boolean
    - .SELF\_CONNECTABLE : Boolean
    - .CONNECT\_CLONEABLE : Boolean
    - .CONNECT\_REQUIRED : Boolean

- .CONNECT\_STYLE\_CHANGE : Boolean
  - .DELETABLE : Boolean
  - .LABEL\_EDITABLE : Boolean
  - .data : Object
  - .textList : Array
  - .clone() ⇒ IShape
  - .createShape() ⇒ \*
- 
- .SpotShape ⇐ GeomShape
    - new OG.shape.SpotShape(label)
    - .TYPE : String
    - .SHAPE\_ID : String
    - .geom : Geometry
    - .label : String
    - .isCollapsed : Boolean
    - .SELECTABLE : Boolean
    - .MOVABLE : Boolean
    - .RESIZABLE : Boolean
    - .CONNECTABLE : Boolean
    - .ENABLE\_FROM : Boolean
    - .ENABLE\_TO : Boolean
    - .SELF\_CONNECTABLE : Boolean
    - .CONNECT\_CLONEABLE : Boolean
    - .CONNECT\_REQUIRED : Boolean
    - .CONNECT\_STYLE\_CHANGE : Boolean
    - .DELETABLE : Boolean

- `.LABEL_EDITABLE` : Boolean
  - `.data` : Object
  - `.textList` : Array
  - `.clone()` ⇒ `IShape`
  - `.createShape()` ⇒ \*
- 
- `.TextShape` ⇐ `IShape`
    - `new OG.shape.TextShape(text)`
    - `.text` : String
    - `.angle` : Number
    - `.TYPE` : String
    - `.SHAPE_ID` : String
    - `.geom` : `Geometry`
    - `.label` : String
    - `.isCollapsed` : Boolean
    - `.SELECTABLE` : Boolean
    - `.MOVABLE` : Boolean
    - `.RESIZABLE` : Boolean
    - `.CONNECTABLE` : Boolean
    - `.ENABLE_FROM` : Boolean
    - `.ENABLE_TO` : Boolean
    - `.SELF_CONNECTABLE` : Boolean
    - `.CONNECT_CLONEABLE` : Boolean
    - `.CONNECT_REQUIRED` : Boolean
    - `.CONNECT_STYLE_CHANGE` : Boolean

- [.DELETABLE](#) : Boolean
- [.LABEL\\_EDITABLE](#) : Boolean
- [.data](#) : Object
- [.textList](#) : Array
- [.createShape\(\)](#) ⇒ \*
- [.clone\(\)](#) ⇒ [IShape](#)
- [.VerticalLaneShape](#) ⇐ [GroupShape](#)
  - [new OG.shape.VerticalLaneShape\(label\)](#)
  - [.GROUP\\_DROPABLE](#) : Boolean
  - [.GROUP\\_COLLAPSIBLE](#) : Boolean
  - [.TYPE](#) : String
  - [.SHAPE\\_ID](#) : String
  - [.geom](#) : [Geometry](#)
  - [.label](#) : String
  - [.isCollapsed](#) : Boolean
  - [.SELECTABLE](#) : Boolean
  - [.MOVABLE](#) : Boolean
  - [.RESIZABLE](#) : Boolean
  - [.CONNECTABLE](#) : Boolean
  - [.ENABLE\\_FROM](#) : Boolean
  - [.ENABLE\\_TO](#) : Boolean
  - [.SELF\\_CONNECTABLE](#) : Boolean
  - [.CONNECT\\_CLONEABLE](#) : Boolean
  - [.CONNECT\\_REQUIRED](#) : Boolean
  - [.CONNECT\\_STYLE\\_CHANGE](#) : Boolean

- .DELETABLE : Boolean
  - .LABEL\_EDITABLE : Boolean
  - .data : Object
  - .textList : Array
  - .createShape() ⇒ \*
  - .clone() ⇒ IShape
- 
- .VerticalPoolShape ⇐ GroupShape
    - new OG.shape.VerticalPoolShape(label)
    - .GROUP\_DROPABLE : Boolean
    - .GROUP\_COLLAPSIBLE : Boolean
    - .TYPE : String
    - .SHAPE\_ID : String
    - .geom : Geometry
    - .label : String
    - .isCollapsed : Boolean
    - .SELECTABLE : Boolean
    - .MOVABLE : Boolean
    - .RESIZABLE : Boolean
    - .CONNECTABLE : Boolean
    - .ENABLE\_FROM : Boolean
    - .ENABLE\_TO : Boolean
    - .SELF\_CONNECTABLE : Boolean
    - .CONNECT\_CLONEABLE : Boolean
    - .CONNECT\_REQUIRED : Boolean

- [.CONNECT\\_STYLE\\_CHANGE](#) : Boolean
- [.DELETABLE](#) : Boolean
- [.LABEL\\_EDITABLE](#) : Boolean
- [.data](#) : Object
- [.textList](#) : Array
- [.createShape\(\)](#) ⇒ \*
- [.clone\(\)](#) ⇒ IShape
  - [.bpnn](#) : object
  - [.elec](#) : object

shape.CircleShape ⇐ [GeomShape](#)

Kind: static class of [shape](#)

Extends: [GeomShape](#)

Requires: module:OG.common.\* , module:OG.geometry.\*

Author: [Seungpil Park \(mailto:spspark@uengine.org\)](mailto:Seungpil_Park@uengine.org)

- [.CircleShape](#) ⇐ [GeomShape](#)
  - [new OG.shape.CircleShape\(label\)](#)
  - [.TYPE](#) : String
  - [.SHAPE\\_ID](#) : String
  - [.geom](#) : [Geometry](#)
  - [.label](#) : String

- .isCollapsed : Boolean
- .SELECTABLE : Boolean
- .MOVABLE : Boolean
- .RESIZABLE : Boolean
- .CONNECTABLE : Boolean
- .ENABLE\_FROM : Boolean
- .ENABLE\_TO : Boolean
- .SELF\_CONNECTABLE : Boolean
- .CONNECT\_CLONEABLE : Boolean
- .CONNECT\_REQUIRED : Boolean
- .CONNECT\_STYLE\_CHANGE : Boolean
- .DELETABLE : Boolean
- .LABEL\_EDITABLE : Boolean
- .data : Object
- .textList : Array
- .clone() ⇒ IShape
- .createShape() ⇒ \*

```
new OG.shape.CircleShape(label)
```

Circle Shape

Param	Type	Description
label	String	라벨 [Optional]

circleShape.TYPE : String

Shape 유형(GEOM, TEXT, HTML, IMAGE, EDGE, GROUP)

Kind: instance property of [CircleShape](#)

circleShape.SHAPE\_ID : String

Shape 을 구분하는 Shape ID(Shape 클래스명과 일치)

Kind: instance property of [CircleShape](#)

Overrides: [SHAPE\\_ID](#)

circleShape.geom : [Geometry](#)

Shape 모양을 나타내는 공간기하 객체(Geometry)

Kind: instance property of [CircleShape](#)

Overrides: [geom](#)

circleShape.label : String

Shape 라벨 텍스트

Kind: instance property of [CircleShape](#)

Overrides: [label](#)

```
circleShape.isCollapsed : Boolean
```

Shape 의 Collapse 여부

Kind: instance property of [CircleShape](#)

```
circleShape.SELECTABLE : Boolean
```

선택 가능여부

Kind: instance property of [CircleShape](#)

```
circleShape.MOVABLE : Boolean
```

이동 가능여부

Kind: instance property of [CircleShape](#)

```
circleShape.RESIZABLE : Boolean
```

리사이즈 가능여부

Kind: instance property of [CircleShape](#)

```
circleShape.CONNECTABLE : Boolean
```

연결 가능여부

Kind: instance property of [CircleShape](#)

circleShape.ENABLE\_FROM : Boolean

From 연결 가능여부 (From(Shape) => To)

Kind: instance property of [CircleShape](#)

circleShape.ENABLE\_TO : Boolean

To 연결 가능여부 (From => To(Shape))

Kind: instance property of [CircleShape](#)

circleShape.SELF\_CONNECTABLE : Boolean

Self 연결 가능여부

Kind: instance property of [CircleShape](#)

circleShape.CONNECT\_CLONEABLE : Boolean

가이드에 자기자신을 복사하는 컨트롤러 여부.

Kind: instance property of [CircleShape](#)

```
circleShape.CONNECT_REQUIRED : Boolean
```

드래그하여 연결시 연결대상 있는 경우에만 Edge 드로잉 처리 여부

Kind: instance property of [CircleShape](#)

```
circleShape.CONNECT_STYLE_CHANGE : Boolean
```

드래그하여 연결시 그룹을 건너뛸때 스타일 변경 여부

Kind: instance property of [CircleShape](#)

```
circleShape.DELETEABLE : Boolean
```

가이드에 삭제 컨트롤러 여부

Kind: instance property of [CircleShape](#)

```
circleShape.LABEL_EDITABLE : Boolean
```

라벨 수정여부

Kind: instance property of [CircleShape](#)

```
circleShape.data : Object
```

도형의 데이터

Kind: instance property of [CircleShape](#)

circleShape.textList : Array

도형 선연결시 선연결 컨트롤러 목록

Kind: instance property of [CircleShape](#)

circleShape.clone() ⇒ [IShape](#)

Shape 을 복사하여 새로운 인스턴스로 반환한다.

Kind: instance method of [CircleShape](#)

Returns: [IShape](#) - 복사된 인스턴스

circleShape.createShape() ⇒ \*

드로잉할 Shape 를 생성하여 반환한다.

Kind: instance method of [CircleShape](#)

Overrides: [createShape](#)

Returns: \* - Shape 정보

shape.EdgeShape ← [IShape](#)

Kind: static class of [shape](#)

Extends: [IShape](#)

Requires: module:OG.common.\* , module:OG.geometry.\*

Author: [Seungpil Park \(mailto:spspark@uengine.org\)](mailto:Seungpil.Park@uengine.org)

- [.EdgeShape](#) <= [IShape](#)

- [new OG.shape.EdgeShape\(from, to, label, fromLabel, toLabel\)](#)
- [.from](#) : Array.<Number>
- [.to](#) : Array.<Number>
- [.fromLabel](#) : String
- [.toLabel](#) : String
- [.TYPE](#) : String
- [.SHAPE\\_ID](#) : String
- [.geom](#) : [Geometry](#)
- [.label](#) : String
- [.isCollapsed](#) : Boolean
- [.SELECTABLE](#) : Boolean
- [.MOVABLE](#) : Boolean
- [.RESIZABLE](#) : Boolean
- [.CONNECTABLE](#) : Boolean
- [.ENABLE\\_FROM](#) : Boolean
- [.ENABLE\\_TO](#) : Boolean
- [.SELF\\_CONNECTABLE](#) : Boolean
- [.CONNECT\\_CLONEABLE](#) : Boolean
- [.CONNECT\\_REQUIRED](#) : Boolean

- [.CONNECT\\_STYLE\\_CHANGE](#) : Boolean
- [.DELETABLE](#) : Boolean
- [.LABEL\\_EDITABLE](#) : Boolean
- [.data](#) : Object
- [.textList](#) : Array
- [.createShape\(\)](#) ⇒ \*
- [.clone\(\)](#) ⇒ [IShape](#)

```
new OG.shape.EdgeShape(from, to, label, fromLabel, toLabel)
```

## Edge Shape

Param	Type	Description
from	Array.<Number>	와이어 시작 좌표
to	Array.<Number>	와이어 끝 좌표
label	String	라벨 [Optional]
fromLabel	String	시작점 라벨 [Optional]
toLabel	String	끝점 라벨 [Optional]

```
edgeShape.from : Array.<Number>
```

## Edge 시작 좌표

Kind: instance property of [EdgeShape](#)

```
edgeShape.to : Array.<Number>
```

Edge 끝 좌표

Kind: instance property of [EdgeShape](#)

edgeShape.fromLabel : String

Edge 시작점 라벨

Kind: instance property of [EdgeShape](#)

edgeShape.toLabel : String

Edge 끝점 라벨

Kind: instance property of [EdgeShape](#)

edgeShape.TYPE : String

Shape 유형(GEOM, TEXT, HTML, IMAGE, EDGE, GROUP)

Kind: instance property of [EdgeShape](#)

Overrides: [TYPE](#)

edgeShape.SHAPE\_ID : String

Shape 을 구분하는 Shape ID(Shape 클래스명과 일치)

Kind: instance property of [EdgeShape](#)

Overrides: [SHAPE\\_ID](#)

edgeShape.geom : [Geometry](#)

Shape 모양을 나타내는 공간기하 객체(Geometry)

Kind: instance property of [EdgeShape](#)

Overrides: [geom](#)

edgeShape.label : String

Shape 라벨 텍스트

Kind: instance property of [EdgeShape](#)

Overrides: [label](#)

edgeShape.isCollapsed : Boolean

Shape 의 Collapse 여부

Kind: instance property of [EdgeShape](#)

edgeShape.SELECTABLE : Boolean

선택 가능여부

Kind: instance property of [EdgeShape](#)

edgeShape.MOVABLE : Boolean

이동 가능여부

Kind: instance property of [EdgeShape](#)

edgeShape.RESIZABLE : Boolean

리사이즈 가능여부

Kind: instance property of [EdgeShape](#)

edgeShape.CONNECTABLE : Boolean

연결 가능여부

Kind: instance property of [EdgeShape](#)

edgeShape.ENABLE\_FROM : Boolean

From 연결 가능여부 (From(Shape) => To)

Kind: instance property of [EdgeShape](#)

edgeShape.ENABLE\_TO : Boolean

To 연결 가능여부 (From => To(Shape))

Kind: instance property of [EdgeShape](#)

edgeShape.SELF\_CONNECTABLE : Boolean

Self 연결 가능여부

Kind: instance property of [EdgeShape](#)

edgeShape.CONNECT\_CLONEABLE : Boolean

가이드에 자기자신을 복사하는 컨트롤러 여부.

Kind: instance property of [EdgeShape](#)

edgeShape.CONNECT\_REQUIRED : Boolean

드래그하여 연결시 연결대상 있는 경우에만 Edge 드로잉 처리 여부

Kind: instance property of [EdgeShape](#)

edgeShape.CONNECT\_STYLE\_CHANGE : Boolean

드래그하여 연결시 그룹을 건너뛸때 스타일 변경 여부

Kind: instance property of [EdgeShape](#)

edgeShape.DELETABLE : Boolean

가이드에 삭제 컨트롤러 여부

Kind: instance property of [EdgeShape](#)

edgeShape.LABEL\_EDITABLE : Boolean

라벨 수정여부

Kind: instance property of [EdgeShape](#)

edgeShape.data : Object

도형의 데이터

Kind: instance property of [EdgeShape](#)

edgeShape.textList : Array

도형 선연결시 선연결 컨트롤러 목록

Kind: instance property of [EdgeShape](#)

edgeShape.createShape() => \*

드로잉할 Shape 를 생성하여 반환한다.

Kind: instance method of [EdgeShape](#)

Overrides: [createShape](#)

Returns: \* - Shape 정보

edgeShape.clone() ⇒ [IShape](#)

Shape 을 복사하여 새로운 인스턴스로 반환한다.

Kind: instance method of [EdgeShape](#)

Overrides: [clone](#)

Returns: [IShape](#) - 복사된 인스턴스

shape.EllipseShape ⇐ [GeomShape](#)

Kind: static class of [shape](#)

Extends: [GeomShape](#)

Requires: module:OG.common.\* , module:OG.geometry.\*

Author: [Seungpil Park \(mailto:spspark@uengine.org\)](mailto:Seungpil.Park@uengine.org)

- [.EllipseShape](#) ⇐ [GeomShape](#)

- [new OG.shape.EllipseShape\(label\)](#)
- [.TYPE](#) : String
- [.SHAPE\\_ID](#) : String

- `.geom` : `Geometry`
- `.label` : String
- `.isCollapsed` : Boolean
- `.SELECTABLE` : Boolean
- `.MOVABLE` : Boolean
- `.RESIZABLE` : Boolean
- `.CONNECTABLE` : Boolean
- `.ENABLE_FROM` : Boolean
- `.ENABLE_TO` : Boolean
- `.SELF_CONNECTABLE` : Boolean
- `.CONNECT_CLONEABLE` : Boolean
- `.CONNECT_REQUIRED` : Boolean
- `.CONNECT_STYLE_CHANGE` : Boolean
- `.DELETABLE` : Boolean
- `.LABEL_EDITABLE` : Boolean
- `.data` : Object
- `.textList` : Array
- `.clone()` ⇒ `IShape`
- `.createShape()` ⇒ \*

```
new OG.shape.EllipseShape(label)
```

Ellipse Shape

Param	Type	Description
label	String	라벨 [Optional]

ellipseShape.TYPE : String

Shape 유형(GEOM, TEXT, HTML, IMAGE, EDGE, GROUP)

Kind: instance property of [EllipseShape](#)

ellipseShape.SHAPE\_ID : String

Shape 을 구분하는 Shape ID(Shape 클래스명과 일치)

Kind: instance property of [EllipseShape](#)

Overrides: [SHAPE\\_ID](#)

ellipseShape.geom : [Geometry](#)

Shape 모양을 나타내는 공간기하 객체(Geometry)

Kind: instance property of [EllipseShape](#)

Overrides: [geom](#)

ellipseShape.label : String

Shape 라벨 텍스트

Kind: instance property of [EllipseShape](#)

Overrides: [label](#)

ellipseShape.isCollapsed : Boolean

Shape 의 Collapse 여부

Kind: instance property of [EllipseShape](#)

ellipseShape.SELECTABLE : Boolean

선택 가능여부

Kind: instance property of [EllipseShape](#)

ellipseShape.MOVABLE : Boolean

이동 가능여부

Kind: instance property of [EllipseShape](#)

ellipseShape.RESIZABLE : Boolean

리사이즈 가능여부

Kind: instance property of [EllipseShape](#)

```
ellipseShape.CONNECTABLE : Boolean
```

연결 가능여부

Kind: instance property of [EllipseShape](#)

```
ellipseShape.ENABLE_FROM : Boolean
```

From 연결 가능여부 (From(Shape) => To)

Kind: instance property of [EllipseShape](#)

```
ellipseShape.ENABLE_TO : Boolean
```

To 연결 가능여부 (From => To(Shape))

Kind: instance property of [EllipseShape](#)

```
ellipseShape.SELF_CONNECTABLE : Boolean
```

Self 연결 가능여부

Kind: instance property of [EllipseShape](#)

```
ellipseShape.CONNECT_CLONEABLE : Boolean
```

가이드에 자기자신을 복사하는 컨트롤러 여부.

Kind: instance property of [EllipseShape](#)

ellipseShape.CONNECT\_REQUIRED : Boolean

드래그하여 연결시 연결대상 있는 경우에만 Edge 드로잉 처리 여부

Kind: instance property of [EllipseShape](#)

ellipseShape.CONNECT\_STYLE\_CHANGE : Boolean

드래그하여 연결시 그룹을 건너뛸때 스타일 변경 여부

Kind: instance property of [EllipseShape](#)

ellipseShape.DELETEABLE : Boolean

가이드에 삭제 컨트롤러 여부

Kind: instance property of [EllipseShape](#)

ellipseShape.LABEL\_EDITABLE : Boolean

라벨 수정여부

Kind: instance property of [EllipseShape](#)

```
ellipseShape.data : Object
```

도형의 데이터

Kind: instance property of [EllipseShape](#)

```
ellipseShape.textList : Array
```

도형 선연결시 선연결 컨트롤러 목록

Kind: instance property of [EllipseShape](#)

```
ellipseShape.clone() => IShape
```

Shape 을 복사하여 새로운 인스턴스로 반환한다.

Kind: instance method of [EllipseShape](#)

Returns: [IShape](#) - 복사된 인스턴스

```
ellipseShape.createShape() => *
```

드로잉할 Shape 를 생성하여 반환한다.

Kind: instance method of [EllipseShape](#)

Overrides: [createShape](#)

Returns: \* - Shape 정보

shape.GemShape <= [IShape](#)

Kind: static class of [shape](#)

Extends: [IShape](#)

Requires: module:OG.common.\* , module:OG.geometry.\*

Author: [Seungpil Park \(mailto:sppark@uengine.org\)](mailto:Seungpil.Park@sppark@uengine.org)

- [.GeomShape](#) <= [IShape](#)
  - [new OG.shape.GemShape\(\)](#)
  - [.TYPE](#) : String
  - [.SHAPE\\_ID](#) : String
  - [.geom](#) : [Geometry](#)
  - [.label](#) : String
  - [.isCollapsed](#) : Boolean
  - [.SELECTABLE](#) : Boolean
  - [.MOVABLE](#) : Boolean
  - [.RESIZABLE](#) : Boolean
  - [.CONNECTABLE](#) : Boolean
  - [.ENABLE\\_FROM](#) : Boolean
  - [.ENABLE\\_TO](#) : Boolean
  - [.SELF\\_CONNECTABLE](#) : Boolean
  - [.CONNECT\\_CLONEABLE](#) : Boolean
  - [.CONNECT\\_REQUIRED](#) : Boolean
  - [.CONNECT\\_STYLE\\_CHANGE](#) : Boolean
  - [.DELETABLE](#) : Boolean

- [.LABEL\\_EDITABLE](#) : Boolean
- [.data](#) : Object
- [.textList](#) : Array
- [.createShape\(\)](#) ⇒ \*
- [.clone\(\)](#) ⇒ [IShape](#)

new OG.shape.GemShape()

Geometry Shape

geomShape.TYPE : String

Shape 유형(GEOM, TEXT, HTML, IMAGE, EDGE, GROUP)

Kind: instance property of [GeomShape](#)

Overrides: [TYPE](#)

geomShape.SHPE\_ID : String

Shape 을 구분하는 Shape ID(Shape 클래스명과 일치)

Kind: instance property of [GeomShape](#)

geomShape.geom : [Geometry](#)

Shape 모양을 나타내는 공간기하 객체(Geometry)

Kind: instance property of [GeomShape](#)

geomShape.label : String

Shape 라벨 텍스트

Kind: instance property of [GeomShape](#)

geomShape.isCollapsed : Boolean

Shape 의 Collapse 여부

Kind: instance property of [GeomShape](#)

geomShape.SELECTABLE : Boolean

선택 가능여부

Kind: instance property of [GeomShape](#)

geomShape.MOVABLE : Boolean

이동 가능여부

Kind: instance property of [GeomShape](#)

geomShape.RESIZABLE : Boolean

리사이즈 가능여부

Kind: instance property of [GeomShape](#)

geomShape.CONNECTABLE : Boolean

연결 가능여부

Kind: instance property of [GeomShape](#)

geomShape.ENABLE\_FROM : Boolean

From 연결 가능여부 (From(Shape) => To)

Kind: instance property of [GeomShape](#)

geomShape.ENABLE\_TO : Boolean

To 연결 가능여부 (From => To(Shape))

Kind: instance property of [GeomShape](#)

```
geomShape.SELF_CONNECTABLE : Boolean
```

Self 연결 가능여부

Kind: instance property of [GeomShape](#)

```
geomShape.CONNECT_CLONEABLE : Boolean
```

가이드에 자기자신을 복사하는 컨트롤러 여부.

Kind: instance property of [GeomShape](#)

```
geomShape.CONNECT_REQUIRED : Boolean
```

드래그하여 연결시 연결대상 있는 경우에만 Edge 드로잉 처리 여부

Kind: instance property of [GeomShape](#)

```
geomShape.CONNECT_STYLE_CHANGE : Boolean
```

드래그하여 연결시 그룹을 건너뛸때 스타일 변경 여부

Kind: instance property of [GeomShape](#)

```
geomShape.DELETABLE : Boolean
```

가이드에 삭제 컨트롤러 여부

Kind: instance property of [GeomShape](#)

geomShape.LABEL\_EDITABLE : Boolean

라벨 수정여부

Kind: instance property of [GeomShape](#)

geomShape.data : Object

도형의 데이터

Kind: instance property of [GeomShape](#)

geomShape.textList : Array

도형 선연결시 선연결 컨트롤러 목록

Kind: instance property of [GeomShape](#)

geomShape.createShape() => \*

드로잉할 Shape 를 생성하여 반환한다.

Kind: instance abstract method of [GeomShape](#)

Returns: \* - Shape 정보

geomShape.clone() ⇒ [IShape](#)

Shape 을 복사하여 새로운 인스턴스로 반환한다.

Kind: instance method of [GeomShape](#)

Overrides: [clone](#)

Returns: [IShape](#) - 복사된 인스턴스

shape.GroupShape ⇐ [IShape](#)

Kind: static class of [shape](#)

Extends: [IShape](#)

Requires: module:OG.common.\* , module:OG.geometry.\*

Author: [Seungpil Park \(mailto:spark@uengine.org\)](mailto:Seungpil.Park@uengine.org)

- [.GroupShape](#) ⇐ [IShape](#)
  - [new OG.shape.GroupShape\(label\)](#)
  - [.GROUP\\_DROPABLE](#) : Boolean
  - [.GROUP\\_COLLAPSIBLE](#) : Boolean
  - [.TYPE](#) : String
  - [.SHAPE\\_ID](#) : String
  - [.geom](#) : [Geometry](#)
  - [.label](#) : String
  - [.isCollapsed](#) : Boolean

- .SELECTABLE : Boolean
- .MOVABLE : Boolean
- .RESIZABLE : Boolean
- .CONNECTABLE : Boolean
- .ENABLE\_FROM : Boolean
- .ENABLE\_TO : Boolean
- .SELF\_CONNECTABLE : Boolean
- .CONNECT\_CLONEABLE : Boolean
- .CONNECT\_REQUIRED : Boolean
- .CONNECT\_STYLE\_CHANGE : Boolean
- .DELETABLE : Boolean
- .LABEL\_EDITABLE : Boolean
- .data : Object
- .textList : Array
- .createShape() ⇒ \*
- .clone() ⇒ IShape

```
new OG.shape.GroupShape(label)
```

## Group Shape

Param	Type	Description
label	String	라벨 [Optional]

```
groupShape.GROUP_DROPABLE : Boolean
```

그룹핑 가능여부

Kind: instance property of [GroupShape](#)

```
groupShape.GROUP_COLLAPSIBLE : Boolean
```

최소화 가능여부

Kind: instance property of [GroupShape](#)

```
groupShape.TYPE : String
```

Shape 유형(GEOM, TEXT, HTML, IMAGE, EDGE, GROUP)

Kind: instance property of [GroupShape](#)

Overrides: [TYPE](#)

```
groupShape.SHAPE_ID : String
```

Shape 을 구분하는 Shape ID(Shape 클래스명과 일치)

Kind: instance property of [GroupShape](#)

Overrides: [SHAPE\\_ID](#)

```
groupShape.geom : Geometry
```

Shape 모양을 나타내는 공간기하 객체(Geometry)

Kind: instance property of [GroupShape](#)

Overrides: [geom](#)

groupShape.label : String

Shape 라벨 텍스트

Kind: instance property of [GroupShape](#)

Overrides: [label](#)

groupShape.isCollapsed : Boolean

Shape 의 Collapse 여부

Kind: instance property of [GroupShape](#)

groupShape.SELECTABLE : Boolean

선택 가능여부

Kind: instance property of [GroupShape](#)

groupShape.MOVABLE : Boolean

이동 가능여부

Kind: instance property of [GroupShape](#)

groupShape.RESIZABLE : Boolean

리사이즈 가능여부

Kind: instance property of [GroupShape](#)

groupShape.CONNECTABLE : Boolean

연결 가능여부

Kind: instance property of [GroupShape](#)

Overrides: [CONNECTABLE](#)

groupShape.ENABLE\_FROM : Boolean

From 연결 가능여부 (From(Shape) => To)

Kind: instance property of [GroupShape](#)

groupShape.ENABLE\_TO : Boolean

To 연결 가능여부 (From => To(Shape))

Kind: instance property of [GroupShape](#)

groupShape.SELF\_CONNECTABLE : Boolean

Self 연결 가능여부

Kind: instance property of [GroupShape](#)

Overrides: [SELF\\_CONNECTABLE](#)

groupShape.CONNECT\_CLONEABLE : Boolean

가이드에 자기자신을 복사하는 컨트롤러 여부.

Kind: instance property of [GroupShape](#)

groupShape.CONNECT\_REQUIRED : Boolean

드래그하여 연결시 연결대상 있는 경우에만 Edge 드로잉 처리 여부

Kind: instance property of [GroupShape](#)

groupShape.CONNECT\_STYLE\_CHANGE : Boolean

드래그하여 연결시 그룹을 건너뛸때 스타일 변경 여부

Kind: instance property of [GroupShape](#)

groupShape.DELETEABLE : Boolean

가이드에 삭제 컨트롤러 여부

Kind: instance property of [GroupShape](#)

groupShape.LABEL\_EDITABLE : Boolean

라벨 수정여부

Kind: instance property of [GroupShape](#)

groupShape.data : Object

도형의 데이터

Kind: instance property of [GroupShape](#)

groupShape.textList : Array

도형 선연결시 선연결 컨트롤러 목록

Kind: instance property of [GroupShape](#)

groupShape.createShape() => \*

드로잉할 Shape 를 생성하여 반환한다.

Kind: instance method of [GroupShape](#)

Overrides: [createShape](#)

Returns: \* - Shape 정보

groupShape.clone() ⇒ [IShape](#)

Shape 을 복사하여 새로운 인스턴스로 반환한다.

Kind: instance method of [GroupShape](#)

Overrides: [clone](#)

Returns: [IShape](#) - 복사된 인스턴스

shape.HorizontalLaneShape ← [GroupShape](#)

Kind: static class of [shape](#)

Extends: [GroupShape](#)

Requires: module:OG.common.\* , module:OG.geometry.\*

Author: [Seungpil Park \(mailto:spark@uengine.org\)](mailto:Seungpil_Park@uengine.org)

- [.HorizontalLaneShape](#) ← [GroupShape](#)

- [new OG.shape.HorizontalLaneShape\(label\)](#)
- [.GROUP\\_DROPABLE](#) : Boolean
- [.GROUP\\_COLLAPSIBLE](#) : Boolean
- [.TYPE](#) : String

- `.SHAPE_ID` : String
- `.geom` : `Geometry`
- `.label` : String
- `.isCollapsed` : Boolean
- `.SELECTABLE` : Boolean
- `.MOVABLE` : Boolean
- `.RESIZABLE` : Boolean
- `.CONNECTABLE` : Boolean
- `.ENABLE_FROM` : Boolean
- `.ENABLE_TO` : Boolean
- `.SELF_CONNECTABLE` : Boolean
- `.CONNECT_CLONEABLE` : Boolean
- `.CONNECT_REQUIRED` : Boolean
- `.CONNECT_STYLE_CHANGE` : Boolean
- `.DELETABLE` : Boolean
- `.LABEL_EDITABLE` : Boolean
- `.data` : Object
- `.textList` : Array
- `.createShape()` ⇒ \*
- `.clone()` ⇒ `IShape`

```
new OG.shape.HorizontalLaneShape(label)
```

Horizontal Swimlane Shape

Param Type Description  
label String 라벨 [Optional]

horizontalLaneShape.GROUP\_DROPABLE : Boolean

그룹핑 가능여부

Kind: instance property of [HorizontalLaneShape](#)

horizontalLaneShape.GROUP\_COLLAPSIBLE : Boolean

최소화 가능여부

Kind: instance property of [HorizontalLaneShape](#)

horizontalLaneShape.TYPE : String

Shape 유형(GEOM, TEXT, HTML, IMAGE, EDGE, GROUP)

Kind: instance property of [HorizontalLaneShape](#)

horizontalLaneShape.SHAPE\_ID : String

Shape 을 구분하는 Shape ID(Shape 클래스명과 일치)

Kind: instance property of [HorizontalLaneShape](#)

Overrides: [SHAPE\\_ID](#)

horizontalLaneShape.geom : [Geometry](#)

Shape 모양을 나타내는 공간기하 객체(Geometry)

Kind: instance property of [HorizontalLaneShape](#)

Overrides: [geom](#)

horizontalLaneShape.label : String

Shape 라벨 텍스트

Kind: instance property of [HorizontalLaneShape](#)

horizontalLaneShape.isCollapsed : Boolean

Shape 의 Collapse 여부

Kind: instance property of [HorizontalLaneShape](#)

horizontalLaneShape.SELECTABLE : Boolean

선택 가능여부

Kind: instance property of [HorizontalLaneShape](#)

```
horizontalLaneShape.MOVABLE : Boolean
```

이동 가능여부

Kind: instance property of [HorizontalLaneShape](#)

```
horizontalLaneShape.RESIZABLE : Boolean
```

리사이즈 가능여부

Kind: instance property of [HorizontalLaneShape](#)

```
horizontalLaneShape.CONNECTABLE : Boolean
```

연결 가능여부

Kind: instance property of [HorizontalLaneShape](#)

```
horizontalLaneShape.ENABLE_FROM : Boolean
```

From 연결 가능여부 (From(Shape) => To)

Kind: instance property of [HorizontalLaneShape](#)

```
horizontalLaneShape.ENABLE_TO : Boolean
```

To 연결 가능여부 (From => To(Shape))

Kind: instance property of [HorizontalLaneShape](#)

horizontalLaneShape.SELF\_CONNECTABLE : Boolean

Self 연결 가능여부

Kind: instance property of [HorizontalLaneShape](#)

horizontalLaneShape.CONNECT\_CLONEABLE : Boolean

가이드에 자기자신을 복사하는 컨트롤러 여부.

Kind: instance property of [HorizontalLaneShape](#)

horizontalLaneShape.CONNECT\_REQUIRED : Boolean

드래그하여 연결시 연결대상 있는 경우에만 Edge 드로잉 처리 여부

Kind: instance property of [HorizontalLaneShape](#)

horizontalLaneShape.CONNECT\_STYLE\_CHANGE : Boolean

드래그하여 연결시 그룹을 건너뛸때 스타일 변경 여부

Kind: instance property of [HorizontalLaneShape](#)

```
horizontalLaneShape.DELETEABLE : Boolean
```

가이드에 삭제 컨트롤러 여부

Kind: instance property of [HorizontalLaneShape](#)

```
horizontalLaneShape.LABEL_EDITABLE : Boolean
```

라벨 수정여부

Kind: instance property of [HorizontalLaneShape](#)

```
horizontalLaneShape.data : Object
```

도형의 데이터

Kind: instance property of [HorizontalLaneShape](#)

```
horizontalLaneShape.textList : Array
```

도형 선연결시 선연결 컨트롤러 목록

Kind: instance property of [HorizontalLaneShape](#)

```
horizontalLaneShape.createShape() => *
```

드로잉할 Shape 를 생성하여 반환한다.

Kind: instance method of [HorizontalLaneShape](#)

Overrides: [createShape](#)

Returns: \* - Shape 정보

horizontalLaneShape.clone() ⇒ [IShape](#)

Shape 을 복사하여 새로운 인스턴스로 반환한다.

Kind: instance method of [HorizontalLaneShape](#)

Returns: [IShape](#) - 복사된 인스턴스

shape.HorizontalPoolShape ← [GroupShape](#)

Kind: static class of [shape](#)

Extends: [GroupShape](#)

Requires: module:OG.common.\* , module:OG.geometry.\*

Author: [Seungpil Park \(mailto:spark@uengine.org\)](mailto:Seungpil.Park@uengine.org)

- [.HorizontalPoolShape](#) ← [GroupShape](#)

- [new OG.shape.HorizontalPoolShape\(label\)](#)
- [.GROUP\\_DROPABLE](#) : Boolean
- [.GROUP\\_COLLAPSIBLE](#) : Boolean
- [.TYPE](#) : String
- [.SHAPE\\_ID](#) : String
- [.geom](#) : [Geometry](#)

- .label : String
- .isCollapsed : Boolean
- .SELECTABLE : Boolean
- .MOVABLE : Boolean
- .RESIZABLE : Boolean
- .CONNECTABLE : Boolean
- .ENABLE\_FROM : Boolean
- .ENABLE\_TO : Boolean
- .SELF\_CONNECTABLE : Boolean
- .CONNECT\_CLONEABLE : Boolean
- .CONNECT\_REQUIRED : Boolean
- .CONNECT\_STYLE\_CHANGE : Boolean
- .DELETABLE : Boolean
- .LABEL\_EDITABLE : Boolean
- .data : Object
- .textList : Array
- .createShape() ⇒ \*
- .clone() ⇒ IShape

```
new OG.shape.HorizontalPoolShape(label)
```

Horizontal Pool Shape

Param	Type	Description
label	String	라벨 [Optional]

horizontalPoolShape.GROUP\_DROPABLE : Boolean

그룹핑 가능여부

Kind: instance property of [HorizontalPoolShape](#)

horizontalPoolShape.GROUP\_COLLAPSIBLE : Boolean

최소화 가능여부

Kind: instance property of [HorizontalPoolShape](#)

Overrides: [GROUP\\_COLLAPSIBLE](#)

horizontalPoolShape.TYPE : String

Shape 유형(GEOM, TEXT, HTML, IMAGE, EDGE, GROUP)

Kind: instance property of [HorizontalPoolShape](#)

horizontalPoolShape.SHAPE\_ID : String

Shape 을 구분하는 Shape ID(Shape 클래스명과 일치)

Kind: instance property of [HorizontalPoolShape](#)

Overrides: [SHAPE\\_ID](#)

horizontalPoolShape.geom : [Geometry](#)

Shape 모양을 나타내는 공간기하 객체(Geometry)

Kind: instance property of [HorizontalPoolShape](#)

Overrides: [geom](#)

horizontalPoolShape.label : String

Shape 라벨 텍스트

Kind: instance property of [HorizontalPoolShape](#)

Overrides: [label](#)

horizontalPoolShape.isCollapsed : Boolean

Shape 의 Collapse 여부

Kind: instance property of [HorizontalPoolShape](#)

horizontalPoolShape.SELECTABLE : Boolean

선택 가능여부

Kind: instance property of [HorizontalPoolShape](#)

```
horizontalPoolShape.MOVABLE : Boolean
```

이동 가능여부

Kind: instance property of [HorizontalPoolShape](#)

```
horizontalPoolShape.RESIZABLE : Boolean
```

리사이즈 가능여부

Kind: instance property of [HorizontalPoolShape](#)

```
horizontalPoolShape.CONNECTABLE : Boolean
```

연결 가능여부

Kind: instance property of [HorizontalPoolShape](#)

Overrides: [CONNECTABLE](#)

```
horizontalPoolShape.ENABLE_FROM : Boolean
```

From 연결 가능여부 (From(Shape) => To)

Kind: instance property of [HorizontalPoolShape](#)

```
horizontalPoolShape.ENABLE_TO : Boolean
```

To 연결 가능여부 (From => To(Shape))

Kind: instance property of [HorizontalPoolShape](#)

horizontalPoolShape.SELF\_CONNECTABLE : Boolean

Self 연결 가능여부

Kind: instance property of [HorizontalPoolShape](#)

horizontalPoolShape.CONNECT\_CLONEABLE : Boolean

가이드에 자기자신을 복사하는 컨트롤러 여부.

Kind: instance property of [HorizontalPoolShape](#)

horizontalPoolShape.CONNECT\_REQUIRED : Boolean

드래그하여 연결시 연결대상 있는 경우에만 Edge 드로잉 처리 여부

Kind: instance property of [HorizontalPoolShape](#)

horizontalPoolShape.CONNECT\_STYLE\_CHANGE : Boolean

드래그하여 연결시 그룹을 건너뛸때 스타일 변경 여부

Kind: instance property of [HorizontalPoolShape](#)

```
horizontalPoolShape.DELETEABLE : Boolean
```

가이드에 삭제 컨트롤러 여부

Kind: instance property of [HorizontalPoolShape](#)

```
horizontalPoolShape.LABEL_EDITABLE : Boolean
```

라벨 수정여부

Kind: instance property of [HorizontalPoolShape](#)

```
horizontalPoolShape.data : Object
```

도형의 데이터

Kind: instance property of [HorizontalPoolShape](#)

```
horizontalPoolShape.textList : Array
```

도형 선연결시 선연결 컨트롤러 목록

Kind: instance property of [HorizontalPoolShape](#)

```
horizontalPoolShape.createShape() => *
```

드로잉할 Shape 를 생성하여 반환한다.

Kind: instance method of [HorizontalPoolShape](#)

Overrides: [createShape](#)

Returns: \* - Shape 정보

horizontalPoolShape.clone() ⇒ [IShape](#)

Shape 을 복사하여 새로운 인스턴스로 반환한다.

Kind: instance method of [HorizontalPoolShape](#)

Returns: [IShape](#) - 복사된 인스턴스

shape.HtmlShape ⇌ [IShape](#)

Kind: static class of [shape](#)

Extends: [IShape](#)

Requires: module:OG.common.\* , module:OG.geometry.\*

Author: [Seungpil Park \(mailto:spspark@uengine.org\)](mailto:Seungpil.Park@uengine.org)

- [.HtmlShape](#) ⇌ [IShape](#)

- [new OG.shape.HtmlShape\(html, label\)](#)
- [.html](#) : String
- [.angle](#) : Number
- [.TYPE](#) : String
- [.SHAPE\\_ID](#) : String

- .geom : Geometry
- .label : String
- .isCollapsed : Boolean
- .SELECTABLE : Boolean
- .MOVABLE : Boolean
- .RESIZABLE : Boolean
- .CONNECTABLE : Boolean
- .ENABLE\_FROM : Boolean
- .ENABLE\_TO : Boolean
- .SELF\_CONNECTABLE : Boolean
- .CONNECT\_CLONEABLE : Boolean
- .CONNECT\_REQUIRED : Boolean
- .CONNECT\_STYLE\_CHANGE : Boolean
- .DELETABLE : Boolean
- .LABEL\_EDITABLE : Boolean
- .data : Object
- .textList : Array
- .createShape() ⇒ \*
- .clone() ⇒ IShape

```
new OG.shape.HtmlShape(html, label)
```

ForeignObject HTML Shape

Param	Type	Description
html	String	임베드 HTML String
label	String	라벨 [Optional]

htmlShape.html : String

드로잉할 임베드 HTML String

Kind: instance property of [HtmlShape](#)

htmlShape.angle : Number

회전각도

Kind: instance property of [HtmlShape](#)

htmlShape.TYPE : String

Shape 유형(GEOM, TEXT, HTML, IMAGE, EDGE, GROUP)

Kind: instance property of [HtmlShape](#)

Overrides: [TYPE](#)

htmlShape.SHAPE\_ID : String

Shape 을 구분하는 Shape ID(Shape 클래스명과 일치)

Kind: instance property of [HtmlShape](#)

Overrides: [SHAPE\\_ID](#)

htmlShape.geom : [Geometry](#)

Shape 모양을 나타내는 공간기하 객체(Geometry)

Kind: instance property of [HtmlShape](#)

htmlShape.label : String

Shape 라벨 텍스트

Kind: instance property of [HtmlShape](#)

Overrides: [label](#)

htmlShape.isCollapsed : Boolean

Shape 의 Collapse 여부

Kind: instance property of [HtmlShape](#)

htmlShape.SELECTABLE : Boolean

선택 가능여부

Kind: instance property of [HtmlShape](#)

htmlShape.MOVABLE : Boolean

이동 가능여부

Kind: instance property of [HtmlShape](#)

htmlShape.RESIZABLE : Boolean

리사이즈 가능여부

Kind: instance property of [HtmlShape](#)

htmlShape.CONNECTABLE : Boolean

연결 가능여부

Kind: instance property of [HtmlShape](#)

htmlShape.ENABLE\_FROM : Boolean

From 연결 가능여부 (From(Shape) => To)

Kind: instance property of [HtmlShape](#)

htmlShape.ENABLE\_TO : Boolean

To 연결 가능여부 (From => To(Shape))

Kind: instance property of [HtmlShape](#)

htmlShape.SELF\_CONNECTABLE : Boolean

Self 연결 가능여부

Kind: instance property of [HtmlShape](#)

htmlShape.CONNECT\_CLONEABLE : Boolean

가이드에 자기자신을 복사하는 컨트롤러 여부.

Kind: instance property of [HtmlShape](#)

htmlShape.CONNECT\_REQUIRED : Boolean

드래그하여 연결시 연결대상 있는 경우에만 Edge 드로잉 처리 여부

Kind: instance property of [HtmlShape](#)

htmlShape.CONNECT\_STYLE\_CHANGE : Boolean

드래그하여 연결시 그룹을 건너뛸때 스타일 변경 여부

Kind: instance property of [HtmlShape](#)

htmlShape.DELETABLE : Boolean

가이드에 삭제 컨트롤러 여부

Kind: instance property of [HtmlShape](#)

htmlShape.LABEL\_EDITABLE : Boolean

라벨 수정여부

Kind: instance property of [HtmlShape](#)

htmlShape.data : Object

도형의 데이터

Kind: instance property of [HtmlShape](#)

htmlShape.textList : Array

도형 선연결시 선연결 컨트롤러 목록

Kind: instance property of [HtmlShape](#)

htmlShape.createShape() => \*

드로잉할 Shape 를 생성하여 반환한다.

Kind: instance method of [HtmlShape](#)

Overrides: [createShape](#)

Returns: \* - Shape 정보

htmlShape.clone() ⇒ [IShape](#)

Shape 을 복사하여 새로운 인스턴스로 반환한다.

Kind: instance method of [HtmlShape](#)

Overrides: [clone](#)

Returns: [IShape](#) - 복사된 인스턴스

shape.IShape

Kind: static class of [shape](#)

Requires: module:OG.common.\* , module:OG.geometry.\*

Author: [Seungpil Park \(mailto:spspark@uengine.org\)](mailto:Seungpil.Park@uengine.org)

- [.IShape](#)

- [new OG.shape.IShape\(\)](#)
- [.TYPE](#) : String
- [.SHAPE\\_ID](#) : String
- [.geom](#) : [Geometry](#)

- .label : String
- .isCollapsed : Boolean
- .SELECTABLE : Boolean
- .MOVABLE : Boolean
- .RESIZABLE : Boolean
- .CONNECTABLE : Boolean
- .ENABLE\_FROM : Boolean
- .ENABLE\_TO : Boolean
- .SELF\_CONNECTABLE : Boolean
- .CONNECT\_CLONEABLE : Boolean
- .CONNECT\_REQUIRED : Boolean
- .CONNECT\_STYLE\_CHANGE : Boolean
- .DELETABLE : Boolean
- .LABEL\_EDITABLE : Boolean
- .data : Object
- .textList : Array
- .createShape() ⇒ \*
- .clone() ⇒ IShape

new OG.shape.IShape()

도형, 텍스트, 이미지 등의 드로잉 될 Object 의 정보를 저장하는 Shape 정보 최상위 인터페이스

iShape.TYPE : String

Shape 유형(GEOM, TEXT, HTML, IMAGE, EDGE, GROUP)

Kind: instance property of [IShape](#)

iShape.SHAPE\_ID : String

Shape 을 구분하는 Shape ID(Shape 클래스명과 일치)

Kind: instance property of [IShape](#)

iShape.geom : [Geometry](#)

Shape 모양을 나타내는 공간기하 객체(Geometry)

Kind: instance property of [IShape](#)

iShape.label : String

Shape 라벨 텍스트

Kind: instance property of [IShape](#)

iShape.isCollapsed : Boolean

Shape 의 Collapse 여부

Kind: instance property of [IShape](#)

iShape.SELECTABLE : Boolean

선택 가능여부

Kind: instance property of [IShape](#)

iShape.MOVABLE : Boolean

이동 가능여부

Kind: instance property of [IShape](#)

iShape.RESIZABLE : Boolean

리사이즈 가능여부

Kind: instance property of [IShape](#)

iShape.CONNECTABLE : Boolean

연결 가능여부

Kind: instance property of [IShape](#)

iShape.ENABLE\_FROM : Boolean

From 연결 가능여부 (From(Shape) => To)

Kind: instance property of [IShape](#)

iShape.ENABLE\_TO : Boolean

To 연결 가능여부 (From => To(Shape))

Kind: instance property of [IShape](#)

iShape.SELF\_CONNECTABLE : Boolean

Self 연결 가능여부

Kind: instance property of [IShape](#)

iShape.CONNECT\_CLONEABLE : Boolean

가이드에 자기자신을 복사하는 컨트롤러 여부.

Kind: instance property of [IShape](#)

iShape.CONNECT\_REQUIRED : Boolean

드래그하여 연결시 연결대상 있는 경우에만 Edge 드로잉 처리 여부

Kind: instance property of [IShape](#)

iShape.CONNECT\_STYLE\_CHANGE : Boolean

드래그하여 연결시 그룹을 건너뛸때 스타일 변경 여부

Kind: instance property of [IShape](#)

iShape.DELETEABLE : Boolean

가이드에 삭제 컨트롤러 여부

Kind: instance property of [IShape](#)

iShape.LABEL\_EDITABLE : Boolean

라벨 수정여부

Kind: instance property of [IShape](#)

iShape.data : Object

도형의 데이터

Kind: instance property of [IShape](#)

*iShape.textList* : Array

도형 선연결시 선연결 컨트롤러 목록

Kind: instance property of [IShape](#)

*iShape.createShape()* ⇒ \*

드로잉할 Shape 를 생성하여 반환한다.

Kind: instance abstract method of [IShape](#)

Returns: \* - Shape 정보

*iShape.clone()* ⇒ [IShape](#)

Shape 을 복사하여 새로운 인스턴스로 반환한다.

Kind: instance abstract method of [IShape](#)

Returns: [IShape](#) - 복사된 인스턴스

*shape.ImageShape* ⇌ [IShape](#)

Kind: static class of [shape](#)

Extends: [IShape](#)

Requires: module:OG.common.\* , module:OG.geometry.\*

Author: [Seungpil Park \(mailto:sppark@uengine.org\)](mailto:Seungpil.Park@sppark@uengine.org)

- `.ImageShape`  $\Leftarrow$  `IShape`
  - `new OG.shape.ImageShape(image, label)`
  - `.image` : String
  - `.angle` : Number
  - `.TYPE` : String
  - `.SHAPE_ID` : String
  - `.geom` : `Geometry`
  - `.label` : String
  - `.isCollapsed` : Boolean
  - `.SELECTABLE` : Boolean
  - `.MOVABLE` : Boolean
  - `.RESIZABLE` : Boolean
  - `.CONNECTABLE` : Boolean
  - `.ENABLE_FROM` : Boolean
  - `.ENABLE_TO` : Boolean
  - `.SELF_CONNECTABLE` : Boolean
  - `.CONNECT_CLONEABLE` : Boolean
  - `.CONNECT_REQUIRED` : Boolean
  - `.CONNECT_STYLE_CHANGE` : Boolean
  - `.DELETABLE` : Boolean
  - `.LABEL_EDITABLE` : Boolean
  - `.data` : Object
  - `.textList` : Array
  - `.createShape()`  $\Rightarrow$  \*
  - `.clone()`  $\Rightarrow$  `IShape`

```
new OG.shape.ImageShape(image, label)
```

## Image Shape

Param Type Description

image String 이미지 URL

label String 라벨 [Optional]

imageShape.image : String

드로잉할 이미지 URL

Kind: instance property of [ImageShape](#)

imageShape.angle : Number

회전각도

Kind: instance property of [ImageShape](#)

imageShape.TYPE : String

Shape 유형(GEOM, TEXT, HTML, IMAGE, EDGE, GROUP)

Kind: instance property of [ImageShape](#)

Overrides: [TYPE](#)

imageShape.SHAPES\_ID : String

Shape 을 구분하는 Shape ID(Shape 클래스명과 일치)

Kind: instance property of [ImageShape](#)

Overrides: [SHAPE\\_ID](#)

imageShape.geom : [Geometry](#)

Shape 모양을 나타내는 공간기하 객체(Geometry)

Kind: instance property of [ImageShape](#)

imageShape.label : String

Shape 라벨 텍스트

Kind: instance property of [ImageShape](#)

Overrides: [label](#)

imageShape.isCollapsed : Boolean

Shape 의 Collapse 여부

Kind: instance property of [ImageShape](#)

imageShape.SELECTABLE : Boolean

선택 가능여부

Kind: instance property of [ImageShape](#)

imageShape.MOVABLE : Boolean

이동 가능여부

Kind: instance property of [ImageShape](#)

imageShape.RESIZABLE : Boolean

리사이즈 가능여부

Kind: instance property of [ImageShape](#)

imageShape.CONNECTABLE : Boolean

연결 가능여부

Kind: instance property of [ImageShape](#)

imageShape.ENABLE\_FROM : Boolean

From 연결 가능여부 (From(Shape) => To)

Kind: instance property of [ImageShape](#)

imageShape.ENABLE\_TO : Boolean

To 연결 가능여부 (From => To(Shape))

Kind: instance property of [ImageShape](#)

imageShape.SELF\_CONNECTABLE : Boolean

Self 연결 가능여부

Kind: instance property of [ImageShape](#)

imageShape.CONNECT\_CLONEABLE : Boolean

가이드에 자기자신을 복사하는 컨트롤러 여부.

Kind: instance property of [ImageShape](#)

imageShape.CONNECT\_REQUIRED : Boolean

드래그하여 연결시 연결대상 있는 경우에만 Edge 드로잉 처리 여부

Kind: instance property of [ImageShape](#)

```
imageShape.CONNECT_STYLE_CHANGE : Boolean
```

드래그하여 연결시 그룹을 건너뛸때 스타일 변경 여부

Kind: instance property of [ImageShape](#)

```
imageShape.DELETEABLE : Boolean
```

가이드에 삭제 컨트롤러 여부

Kind: instance property of [ImageShape](#)

```
imageShape.LABEL_EDITABLE : Boolean
```

라벨 수정여부

Kind: instance property of [ImageShape](#)

```
imageShape.data : Object
```

도형의 데이터

Kind: instance property of [ImageShape](#)

```
imageShape.textList : Array
```

Kind: instance property of [ImageShape](#)

imageShape.createShape() ⇒ \*

드로잉할 Shape 를 생성하여 반환한다.

Kind: instance method of [ImageShape](#)

Overrides: [createShape](#)

Returns: \* - Shape 정보

imageShape.clone() ⇒ [IShape](#)

Shape 을 복사하여 새로운 인스턴스로 반환한다.

Kind: instance method of [ImageShape](#)

Overrides: [clone](#)

Returns: [IShape](#) - 복사된 인스턴스

shape.RectangleShape ⇌ [GeomShape](#)

Kind: static class of [shape](#)

Extends: [GeomShape](#)

Requires: module:OG.common.\* , module:OG.geometry.\*

Author: [Seungpil Park \(mailto:spark@uengine.org\)](mailto:Seungpil.Park@uengine.org)

- `.RectangleShape`  $\Leftarrow$  `GeomShape`
  - `new OG.shape.RectangleShape(label)`
  - `.TYPE` : String
  - `.SHAPE_ID` : String
  - `.geom` : `Geometry`
  - `.label` : String
  - `.isCollapsed` : Boolean
  - `.SELECTABLE` : Boolean
  - `.MOVABLE` : Boolean
  - `.RESIZABLE` : Boolean
  - `.CONNECTABLE` : Boolean
  - `.ENABLE_FROM` : Boolean
  - `.ENABLE_TO` : Boolean
  - `.SELF_CONNECTABLE` : Boolean
  - `.CONNECT_CLONEABLE` : Boolean
  - `.CONNECT_REQUIRED` : Boolean
  - `.CONNECT_STYLE_CHANGE` : Boolean
  - `.DELETABLE` : Boolean
  - `.LABEL_EDITABLE` : Boolean
  - `.data` : Object
  - `.textList` : Array
  - `.clone()`  $\Rightarrow$  `IShape`
  - `.createShape()`  $\Rightarrow$  \*

```
new OG.shape.RectangleShape(label)
```

## Rectangle Shape

Param Type Description

label String 라벨 [Optional]

rectangleShape.TYPE : String

Shape 유형(GEOM, TEXT, HTML, IMAGE, EDGE, GROUP)

Kind: instance property of [RectangleShape](#)

rectangleShape.SHAPE\_ID : String

Shape 을 구분하는 Shape ID(Shape 클래스명과 일치)

Kind: instance property of [RectangleShape](#)

Overrides: [SHAPE\\_ID](#)

rectangleShape.geom : [Geometry](#)

Shape 모양을 나타내는 공간기하 객체(Geometry)

Kind: instance property of [RectangleShape](#)

Overrides: [geom](#)

rectangleShape.label : String

Shape 라벨 텍스트

Kind: instance property of [RectangleShape](#)

Overrides: [label](#)

rectangleShape.isCollapsed : Boolean

Shape 의 Collapse 여부

Kind: instance property of [RectangleShape](#)

rectangleShape.SELECTABLE : Boolean

선택 가능여부

Kind: instance property of [RectangleShape](#)

rectangleShape.MOVABLE : Boolean

이동 가능여부

Kind: instance property of [RectangleShape](#)

rectangleShape.RESIZABLE : Boolean

리사이즈 가능여부

Kind: instance property of [RectangleShape](#)

rectangleShape.CONNECTABLE : Boolean

연결 가능여부

Kind: instance property of [RectangleShape](#)

rectangleShape.ENABLE\_FROM : Boolean

From 연결 가능여부 (From(Shape) => To)

Kind: instance property of [RectangleShape](#)

rectangleShape.ENABLE\_TO : Boolean

To 연결 가능여부 (From => To(Shape))

Kind: instance property of [RectangleShape](#)

rectangleShape.SELF\_CONNECTABLE : Boolean

Self 연결 가능여부

Kind: instance property of [RectangleShape](#)

rectangleShape.CONNECT\_CLONEABLE : Boolean

가이드에 자기자신을 복사하는 컨트롤러 여부.

Kind: instance property of [RectangleShape](#)

rectangleShape.CONNECT\_REQUIRED : Boolean

드래그하여 연결시 연결대상 있는 경우에만 Edge 드로잉 처리 여부

Kind: instance property of [RectangleShape](#)

rectangleShape.CONNECT\_STYLE\_CHANGE : Boolean

드래그하여 연결시 그룹을 건너뛸때 스타일 변경 여부

Kind: instance property of [RectangleShape](#)

rectangleShape.DELETEABLE : Boolean

가이드에 삭제 컨트롤러 여부

Kind: instance property of [RectangleShape](#)

rectangleShape.LABEL\_EDITABLE : Boolean

라벨 수정여부

Kind: instance property of [RectangleShape](#)

rectangleShape.data : Object

도형의 데이터

Kind: instance property of [RectangleShape](#)

rectangleShape.textList : Array

도형 선연결시 선연결 컨트롤러 목록

Kind: instance property of [RectangleShape](#)

rectangleShape.clone() ⇒ [IShape](#)

Shape 을 복사하여 새로운 인스턴스로 반환한다.

Kind: instance method of [RectangleShape](#)

Returns: [IShape](#) - 복사된 인스턴스

rectangleShape.createShape() ⇒ \*

드로잉할 Shape 를 생성하여 반환한다.

Kind: instance method of [RectangleShape](#)

Overrides: [createShape](#)

Returns: \* - Shape 정보

shape.SpotShape ⇐ [GeomShape](#)

Kind: static class of [shape](#)

Extends: [GeomShape](#)

Requires: module:OG.common.\* , module:OG.geometry.\*

Author: [Seungpil Park \(mailto:spark@uengine.org\)](mailto:Seungpil.Park@uengine.org)

- [.SpotShape](#) ⇐ [GeomShape](#)
  - [new OG.shape.SpotShape\(label\)](#)
  - [.TYPE](#) : String
  - [.SHAPE\\_ID](#) : String
  - [.geom](#) : [Geometry](#)
  - [.label](#) : String
  - [.isCollapsed](#) : Boolean
  - [.SELECTABLE](#) : Boolean
  - [.MOVABLE](#) : Boolean
  - [.RESIZABLE](#) : Boolean
  - [.CONNECTABLE](#) : Boolean
  - [.ENABLE\\_FROM](#) : Boolean
  - [.ENABLE\\_TO](#) : Boolean

- .SELF\_CONNECTABLE : Boolean
- .CONNECT\_CLONEABLE : Boolean
- .CONNECT\_REQUIRED : Boolean
- .CONNECT\_STYLE\_CHANGE : Boolean
- .DELETABLE : Boolean
- .LABEL\_EDITABLE : Boolean
- .data : Object
- .textList : Array
- .clone() ⇒ IShape
- .createShape() ⇒ \*

```
new OG.shape.SpotShape(label)
```

SpotShape Shape

Param	Type	Description
label	String	라벨 [Optional]

```
spotShape.TYPE : String
```

Shape 유형(GEOM, TEXT, HTML, IMAGE, EDGE, GROUP)

Kind: instance property of [SpotShape](#)

```
spotShape.SHAPE_ID : String
```

Shape 을 구분하는 Shape ID(Shape 클래스명과 일치)

Kind: instance property of [SpotShape](#)

Overrides: [SHAPE\\_ID](#)

spotShape.geom : [Geometry](#)

Shape 모양을 나타내는 공간기하 객체(Geometry)

Kind: instance property of [SpotShape](#)

Overrides: [geom](#)

spotShape.label : String

Shape 라벨 텍스트

Kind: instance property of [SpotShape](#)

Overrides: [label](#)

spotShape.isCollapsed : Boolean

Shape 의 Collapse 여부

Kind: instance property of [SpotShape](#)

spotShape.SELECTABLE : Boolean

선택 가능여부

Kind: instance property of [SpotShape](#)

spotShape.MOVABLE : Boolean

이동 가능여부

Kind: instance property of [SpotShape](#)

spotShape.RESIZABLE : Boolean

리사이즈 가능여부

Kind: instance property of [SpotShape](#)

spotShape.CONNECTABLE : Boolean

연결 가능여부

Kind: instance property of [SpotShape](#)

spotShape.ENABLE\_FROM : Boolean

From 연결 가능여부 (From(Shape) => To)

Kind: instance property of [SpotShape](#)

spotShape.ENABLE\_TO : Boolean

To 연결 가능여부 (From => To(Shape))

Kind: instance property of [SpotShape](#)

spotShape.SELF\_CONNECTABLE : Boolean

Self 연결 가능여부

Kind: instance property of [SpotShape](#)

spotShape.CONNECT\_CLONEABLE : Boolean

가이드에 자기자신을 복사하는 컨트롤러 여부.

Kind: instance property of [SpotShape](#)

spotShape.CONNECT\_REQUIRED : Boolean

드래그하여 연결시 연결대상 있는 경우에만 Edge 드로잉 처리 여부

Kind: instance property of [SpotShape](#)

spotShape.CONNECT\_STYLE\_CHANGE : Boolean

드래그하여 연결시 그룹을 건너뛸때 스타일 변경 여부

Kind: instance property of [SpotShape](#)

spotShape.DELETABLE : Boolean

가이드에 삭제 컨트롤러 여부

Kind: instance property of [SpotShape](#)

spotShape.LABEL\_EDITABLE : Boolean

라벨 수정여부

Kind: instance property of [SpotShape](#)

spotShape.data : Object

도형의 데이터

Kind: instance property of [SpotShape](#)

spotShape.textList : Array

도형 선연결시 선연결 컨트롤러 목록

Kind: instance property of [SpotShape](#)

spotShape.clone() ⇒ [IShape](#)

Shape 을 복사하여 새로인 인스턴스로 반환한다.

Kind: instance method of [SpotShape](#)

Returns: [IShape](#) - 복사된 인스턴스

spotShape.createShape() ⇒ \*

드로잉할 Shape 를 생성하여 반환한다.

Kind: instance method of [SpotShape](#)

Overrides: [createShape](#)

Returns: \* - Shape 정보

shape.TextShape ⇐ [IShape](#)

Kind: static class of [shape](#)

Extends: [IShape](#)

Requires: module:OG.common.\* , module:OG.geometry.\*

Author: [Seungpil Park \(mailto:spark@uengine.org\)](mailto:Seungpil.Park@uengine.org)

- [.TextShape](#) ⇐ [IShape](#)

- [new OG.shape.TextShape\(text\)](#)

- `.text` : String
- `.angle` : Number
- `.TYPE` : String
- `.SHAPE_ID` : String
- `.geom` : [Geometry](#)
- `.label` : String
- `.isCollapsed` : Boolean
- `.SELECTABLE` : Boolean
- `.MOVABLE` : Boolean
- `.RESIZABLE` : Boolean
- `.CONNECTABLE` : Boolean
- `.ENABLE_FROM` : Boolean
- `.ENABLE_TO` : Boolean
- `.SELF_CONNECTABLE` : Boolean
- `.CONNECT_CLONEABLE` : Boolean
- `.CONNECT_REQUIRED` : Boolean
- `.CONNECT_STYLE_CHANGE` : Boolean
- `.DELETABLE` : Boolean
- `.LABEL_EDITABLE` : Boolean
- `.data` : Object
- `.textList` : Array
- `.createShape()` ⇒ \*
- `.clone()` ⇒ [IShape](#)

```
new OG.shape.TextShape(text)
```

## Text Shape

Param Type Description

text String 텍스트

textShape.text : String

드로잉할 텍스트

Kind: instance property of [TextShape](#)

textShape.angle : Number

회전각도

Kind: instance property of [TextShape](#)

textShape.TYPE : String

Shape 유형(GEOM, TEXT, HTML, IMAGE, EDGE, GROUP)

Kind: instance property of [TextShape](#)

Overrides: [TYPE](#)

textShape.SHAPE\_ID : String

Shape 을 구분하는 Shape ID(Shape 클래스명과 일치)

Kind: instance property of [TextShape](#)

Overrides: [SHAPE\\_ID](#)

textShape.geom : [Geometry](#)

Shape 모양을 나타내는 공간기하 객체(Geometry)

Kind: instance property of [TextShape](#)

textShape.label : String

Shape 라벨 텍스트

Kind: instance property of [TextShape](#)

textShape.isCollapsed : Boolean

Shape 의 Collapse 여부

Kind: instance property of [TextShape](#)

textShape.SELECTABLE : Boolean

선택 가능여부

Kind: instance property of [TextShape](#)

textShape.MOVABLE : Boolean

이동 가능여부

Kind: instance property of [TextShape](#)

textShape.RESIZABLE : Boolean

리사이즈 가능여부

Kind: instance property of [TextShape](#)

textShape.CONNECTABLE : Boolean

연결 가능여부

Kind: instance property of [TextShape](#)

textShape.ENABLE\_FROM : Boolean

From 연결 가능여부 (From(Shape) => To)

Kind: instance property of [TextShape](#)

textShape.ENABLE\_TO : Boolean

To 연결 가능여부 (From => To(Shape))

Kind: instance property of [TextShape](#)

textShape.SELF\_CONNECTABLE : Boolean

Self 연결 가능여부

Kind: instance property of [TextShape](#)

textShape.CONNECT\_CLONEABLE : Boolean

가이드에 자기자신을 복사하는 컨트롤러 여부.

Kind: instance property of [TextShape](#)

textShape.CONNECT\_REQUIRED : Boolean

드래그하여 연결시 연결대상 있는 경우에만 Edge 드로잉 처리 여부

Kind: instance property of [TextShape](#)

textShape.CONNECT\_STYLE\_CHANGE : Boolean

드래그하여 연결시 그룹을 건너뛸때 스타일 변경 여부

Kind: instance property of [TextShape](#)

textShape.DELETABLE : Boolean

가이드에 삭제 컨트롤러 여부

Kind: instance property of [TextShape](#)

textShape.LABEL\_EDITABLE : Boolean

라벨 수정여부

Kind: instance property of [TextShape](#)

textShape.data : Object

도형의 데이터

Kind: instance property of [TextShape](#)

textShape.textList : Array

도형 선연결시 선연결 컨트롤러 목록

Kind: instance property of [TextShape](#)

```
textShape.createShape() => *
```

드로잉할 Shape 를 생성하여 반환한다.

Kind: instance method of [TextShape](#)

Overrides: [createShape](#)

Returns: \* - Shape 정보

```
textShape.clone() => IShape
```

Shape 을 복사하여 새로운 인스턴스로 반환한다.

Kind: instance method of [TextShape](#)

Overrides: [clone](#)

Returns: [IShape](#) - 복사된 인스턴스

```
shape.VerticalLaneShape <= GroupShape
```

Kind: static class of [shape](#)

Extends: [GroupShape](#)

Requires: module:OG.common.\* , module:OG.geometry.\*

Author: [Seungpil Park \(mailto:sppark@uengine.org\)](mailto:Seungpil.Park@sppark@uengine.org)

- [.VerticalLaneShape](#) <= [GroupShape](#)

- [new OG.shape.VerticalLaneShape\(label\)](#)

- [.GROUP\\_DROPABLE](#) : Boolean

- .GROUP\_COLLAPSIBLE : Boolean
- .TYPE : String
- .SHAPE\_ID : String
- .geom : Geometry
- .label : String
- .isCollapsed : Boolean
- .SELECTABLE : Boolean
- .MOVABLE : Boolean
- .RESIZABLE : Boolean
- .CONNECTABLE : Boolean
- .ENABLE\_FROM : Boolean
- .ENABLE\_TO : Boolean
- .SELF\_CONNECTABLE : Boolean
- .CONNECT\_CLONEABLE : Boolean
- .CONNECT\_REQUIRED : Boolean
- .CONNECT\_STYLE\_CHANGE : Boolean
- .DELETABLE : Boolean
- .LABEL\_EDITABLE : Boolean
- .data : Object
- .textList : Array
- .createShape() ⇒ \*
- .clone() ⇒ IShape

```
new OG.shape.VerticalLaneShape(label)
```

Vertical Swimlane Shape

Param Type Description

label String 라벨

verticalLaneShape.GROUP\_DROPABLE : Boolean

그룹핑 가능여부

Kind: instance property of [VerticalLaneShape](#)

verticalLaneShape.GROUP\_COLLAPSIBLE : Boolean

최소화 가능여부

Kind: instance property of [VerticalLaneShape](#)

verticalLaneShape.TYPE : String

Shape 유형(GEOM, TEXT, HTML, IMAGE, EDGE, GROUP)

Kind: instance property of [VerticalLaneShape](#)

verticalLaneShape.SHAPE\_ID : String

Shape 을 구분하는 Shape ID(Shape 클래스명과 일치)

Kind: instance property of [VerticalLaneShape](#)

Overrides: [SHAPE\\_ID](#)

verticalLaneShape.geom : [Geometry](#)

Shape 모양을 나타내는 공간기하 객체(Geometry)

Kind: instance property of [VerticalLaneShape](#)

Overrides: [geom](#)

verticalLaneShape.label : String

Shape 라벨 텍스트

Kind: instance property of [VerticalLaneShape](#)

verticalLaneShape.isCollapsed : Boolean

Shape 의 Collapse 여부

Kind: instance property of [VerticalLaneShape](#)

verticalLaneShape.SELECTABLE : Boolean

선택 가능여부

Kind: instance property of [VerticalLaneShape](#)

verticalLaneShape.MOVABLE : Boolean

이동 가능여부

Kind: instance property of [VerticalLaneShape](#)

verticalLaneShape.RESIZABLE : Boolean

리사이즈 가능여부

Kind: instance property of [VerticalLaneShape](#)

verticalLaneShape.CONNECTABLE : Boolean

연결 가능여부

Kind: instance property of [VerticalLaneShape](#)

verticalLaneShape.ENABLE\_FROM : Boolean

From 연결 가능여부 (From(Shape) => To)

Kind: instance property of [VerticalLaneShape](#)

verticalLaneShape.ENABLE\_TO : Boolean

To 연결 가능여부 (From => To(Shape))

Kind: instance property of [VerticalLaneShape](#)

verticalLaneShape.SELF\_CONNECTABLE : Boolean

Self 연결 가능여부

Kind: instance property of [VerticalLaneShape](#)

verticalLaneShape.CONNECT\_CLONEABLE : Boolean

가이드에 자기자신을 복사하는 컨트롤러 여부.

Kind: instance property of [VerticalLaneShape](#)

verticalLaneShape.CONNECT\_REQUIRED : Boolean

드래그하여 연결시 연결대상 있는 경우에만 Edge 드로잉 처리 여부

Kind: instance property of [VerticalLaneShape](#)

verticalLaneShape.CONNECT\_STYLE\_CHANGE : Boolean

드래그하여 연결시 그룹을 건너뛸때 스타일 변경 여부

Kind: instance property of [VerticalLaneShape](#)

verticalLaneShape.DELETEABLE : Boolean

가이드에 삭제 컨트롤러 여부

Kind: instance property of [VerticalLaneShape](#)

verticalLaneShape.LABEL\_EDITABLE : Boolean

라벨 수정여부

Kind: instance property of [VerticalLaneShape](#)

verticalLaneShape.data : Object

도형의 데이터

Kind: instance property of [VerticalLaneShape](#)

verticalLaneShape.textList : Array

도형 선연결시 선연결 컨트롤러 목록

Kind: instance property of [VerticalLaneShape](#)

verticalLaneShape.createShape() => \*

드로잉할 Shape 를 생성하여 반환한다.

Kind: instance method of [VerticalLaneShape](#)

Overrides: [createShape](#)

Returns: \* - Shape 정보

verticalLaneShape.clone() ⇒ [IShape](#)

Shape 을 복사하여 새로운 인스턴스로 반환한다.

Kind: instance method of [VerticalLaneShape](#)

Returns: [IShape](#) - 복사된 인스턴스

shape.VerticalPoolShape ⇐ [GroupShape](#)

Kind: static class of [shape](#)

Extends: [GroupShape](#)

Requires: module:OG.common.\* , module:OG.geometry.\*

- [.VerticalPoolShape](#) ⇐ [GroupShape](#)

- [new OG.shape.VerticalPoolShape\(label\)](#)
- [.GROUP\\_DROPABLE](#) : Boolean
- [.GROUP\\_COLLAPSIBLE](#) : Boolean
- [.TYPE](#) : String
- [.SHAPE\\_ID](#) : String

- `.geom` : `Geometry`
- `.label` : String
- `.isCollapsed` : Boolean
- `.SELECTABLE` : Boolean
- `.MOVABLE` : Boolean
- `.RESIZABLE` : Boolean
- `.CONNECTABLE` : Boolean
- `.ENABLE_FROM` : Boolean
- `.ENABLE_TO` : Boolean
- `.SELF_CONNECTABLE` : Boolean
- `.CONNECT_CLONEABLE` : Boolean
- `.CONNECT_REQUIRED` : Boolean
- `.CONNECT_STYLE_CHANGE` : Boolean
- `.DELETABLE` : Boolean
- `.LABEL_EDITABLE` : Boolean
- `.data` : Object
- `.textList` : Array
- `.createShape()` ⇒ \*
- `.clone()` ⇒ `IShape`

```
new OG.shape.VerticalPoolShape(label)
```

Vertical Pool Shape

Param Type Description

label String 라벨

verticalPoolShape.GROUP\_DROPABLE : Boolean

그룹핑 가능여부

Kind: instance property of [VerticalPoolShape](#)

verticalPoolShape.GROUP\_COLLAPSIBLE : Boolean

최소화 가능여부

Kind: instance property of [VerticalPoolShape](#)

Overrides: [GROUP\\_COLLAPSIBLE](#)

verticalPoolShape.TYPE : String

Shape 유형(GEOM, TEXT, HTML, IMAGE, EDGE, GROUP)

Kind: instance property of [VerticalPoolShape](#)

verticalPoolShape.SHAPE\_ID : String

Shape 을 구분하는 Shape ID(Shape 클래스명과 일치)

Kind: instance property of [VerticalPoolShape](#)

Overrides: [SHAPE\\_ID](#)

verticalPoolShape.geom : [Geometry](#)

Shape 모양을 나타내는 공간기하 객체(Geometry)

Kind: instance property of [VerticalPoolShape](#)

Overrides: [geom](#)

verticalPoolShape.label : String

Shape 라벨 텍스트

Kind: instance property of [VerticalPoolShape](#)

verticalPoolShape.isCollapsed : Boolean

Shape 의 Collapse 여부

Kind: instance property of [VerticalPoolShape](#)

verticalPoolShape.SELECTABLE : Boolean

선택 가능여부

Kind: instance property of [VerticalPoolShape](#)

verticalPoolShape.MOVABLE : Boolean

이동 가능여부

Kind: instance property of [VerticalPoolShape](#)

verticalPoolShape.RESIZABLE : Boolean

리사이즈 가능여부

Kind: instance property of [VerticalPoolShape](#)

verticalPoolShape.CONNECTABLE : Boolean

연결 가능여부

Kind: instance property of [VerticalPoolShape](#)

Overrides: [CONNECTABLE](#)

verticalPoolShape.ENABLE\_FROM : Boolean

From 연결 가능여부 (From(Shape) => To)

Kind: instance property of [VerticalPoolShape](#)

verticalPoolShape.ENABLE\_T0 : Boolean

To 연결 가능여부 (From => To(Shape))

Kind: instance property of [VerticalPoolShape](#)

verticalPoolShape.SELF\_CONNECTABLE : Boolean

Self 연결 가능여부

Kind: instance property of [VerticalPoolShape](#)

verticalPoolShape.CONNECT\_CLONEABLE : Boolean

가이드에 자기자신을 복사하는 컨트롤러 여부.

Kind: instance property of [VerticalPoolShape](#)

verticalPoolShape.CONNECT\_REQUIRED : Boolean

드래그하여 연결시 연결대상 있는 경우에만 Edge 드로잉 처리 여부

Kind: instance property of [VerticalPoolShape](#)

verticalPoolShape.CONNECT\_STYLE\_CHANGE : Boolean

드래그하여 연결시 그룹을 건너뛸때 스타일 변경 여부

Kind: instance property of [VerticalPoolShape](#)

verticalPoolShape.DELETEABLE : Boolean

가이드에 삭제 컨트롤러 여부

Kind: instance property of [VerticalPoolShape](#)

verticalPoolShape.LABEL\_EDITABLE : Boolean

라벨 수정여부

Kind: instance property of [VerticalPoolShape](#)

verticalPoolShape.data : Object

도형의 데이터

Kind: instance property of [VerticalPoolShape](#)

verticalPoolShape.textList : Array

도형 선연결시 선연결 컨트롤러 목록

Kind: instance property of [VerticalPoolShape](#)

```
verticalPoolShape.createShape() => *
```

드로잉할 Shape 를 생성하여 반환한다.

Kind: instance method of [VerticalPoolShape](#)

Overrides: [createShape](#)

Returns: \* - Shape 정보

```
verticalPoolShape.clone() => IShape
```

Shape 을 복사하여 새로인 인스턴스로 반환한다.

Kind: instance method of [VerticalPoolShape](#)

Returns: [IShape](#) - 복사된 인스턴스

```
shape.bpmn : object
```

Kind: static namespace of [shape](#)

```
shape.elec : object
```

Kind: static namespace of [shape](#)

```
override(origclass, overrides)
```

Adds a list of functions to the prototype of an existing class, overwriting any existing methods with the same name.

Usage:

```
Ext.override(MyClass, {  
    newMethod1: function(){  
        // etc.  
    },  
    newMethod2: function(foo){  
        // etc.  
    }  
});
```

Kind: global function

Param	Type	Description
origClass	Object	The class to override
overrides	Object	The list of functions to add to origClass. This should be specified as an object literal containing one or more methods.