Start Easy

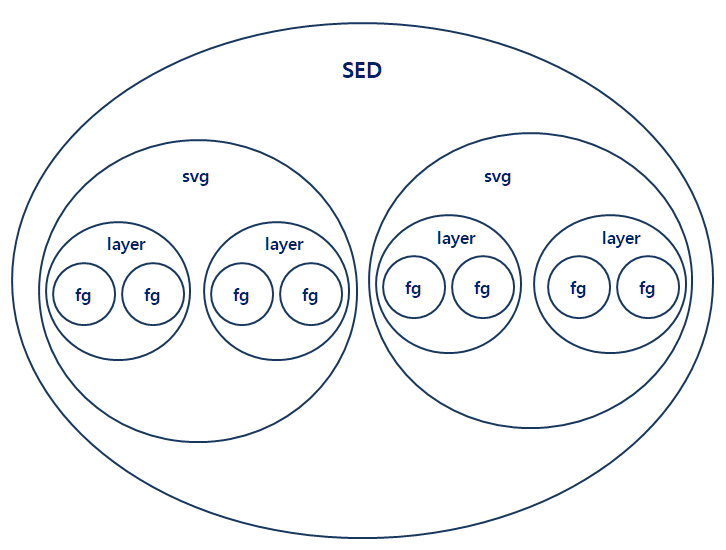
Data Flow Diagram

사용설명서

이병옥

Start Easy Diagram 구성의 이해

Start Easy Diagram은 다음과 같은 구성을 갖는다.



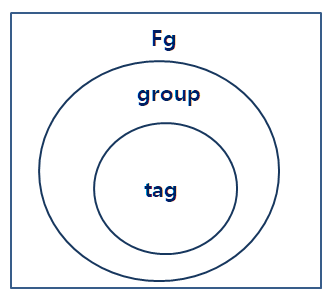
Sed : 스타트이지 다이어그램 객체(전체)

sed.svg : 선택된 svg 엘러먼트

sed.svg.selectedLayer: 선택된 레이어 객체

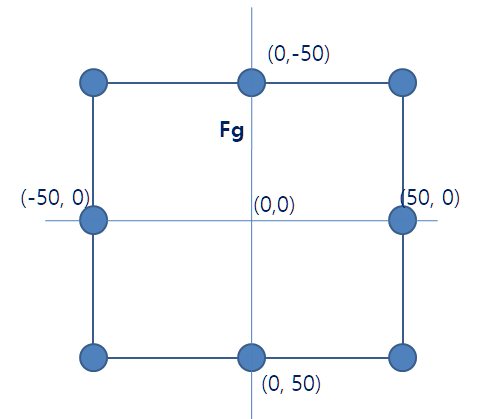
sed.svg.selectedFg: 선택된 Figure 객체

Figure의 구성



Figure는 하나의 도형을 나타내는 객체이고 svg의 group을 멤버객체로 가지고 있다. group의 하위에는 tag 객체가 존재한다. (경에 따라 여러 계층의 group과 tag를 가질 수도 있다.)

객체는 백분율을 좌표로 계산하기 때문에 다음과 같은 백분율로 계산한다.



sed1.svg.selectedFg.moveXY(200, 300) 과 같이 이동할 경우 위 도형의 중심이 200, 300으로 이동하게 된다.

설치 및 설정하기

라이브러리를 [www.starteasy.io](http://www.starteasy.io)에서 다운로드 받는다.

라이브러리 목록

기본라이브러리 sed.zip

데이터 플로우 다이어그램 : flowdiagram.zip

두 개의 압축파일을 풀면 js 파일과 샘플 파일이 나타난다. 플로우 다이어그램의 index.html, index.js, flowdiagram.js, define.js와 img 폴더의 이미지는 사용자가 직접 수정하여 사용할 수 있다. 임시로 만들어진 샘플은 특별히 Data Flow Diagram에 맞추어져 있다.

기본 라이브러리는 압축되어 있으며 사용자가 수정할 수 없다.

먼저 index.html 파일을 살펴보자. 실행하면 라이브러리에서 압축해제한 js파일들이 존재하는데 사용자 환경에 맞추어 경로를 맞추어 주면 된다.

<script language="javascript" src="../../sed/js/se/lib\_flowline.js?\_dc=20181209"></script> <!-- 라이브러리 경로는 맞추어 주어야 한다. -->

<script language="javascript" src="../../sed/js/se/lib.js?\_dc=20181209"></script> <!-- 라이브러리 경로는 맞추어 주어야 한다. -->

<script language="javascript" src="../../sed/js/se/lib\_text.js?\_dc=20181209"></script>

<script language="javascript" src="../../sed/js/se/fg.js?\_dc=20181209"></script>

<script language="javascript" src="../../sed/js/se/fg\_shapes.js?\_dc=20181209"></script>

<script language="javascript" src="../../sed/js/se/sed.js?\_dc=20181209"></script>

<script language="javascript" src="../../sed/js/se/lib\_ex.js?\_dc=20181209"></script>

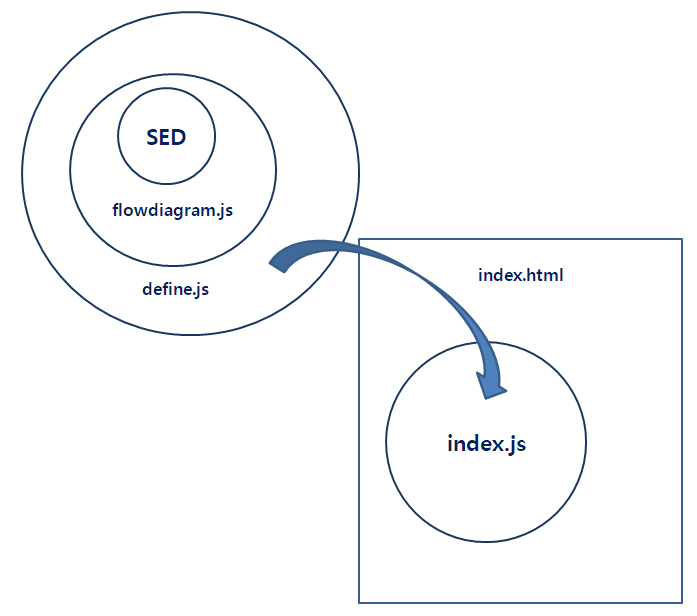
<script language="javascript" src="./flowdiagram.js?\_dc=20181209"></script>

<script language="javascript" src="./define.js?\_dc=20181209"></script>

<script language="javascript" src="./index.js?\_dc=20181209"></script>

\* 경로는 여러분의 PC환경에 따라 다를 수 있다.

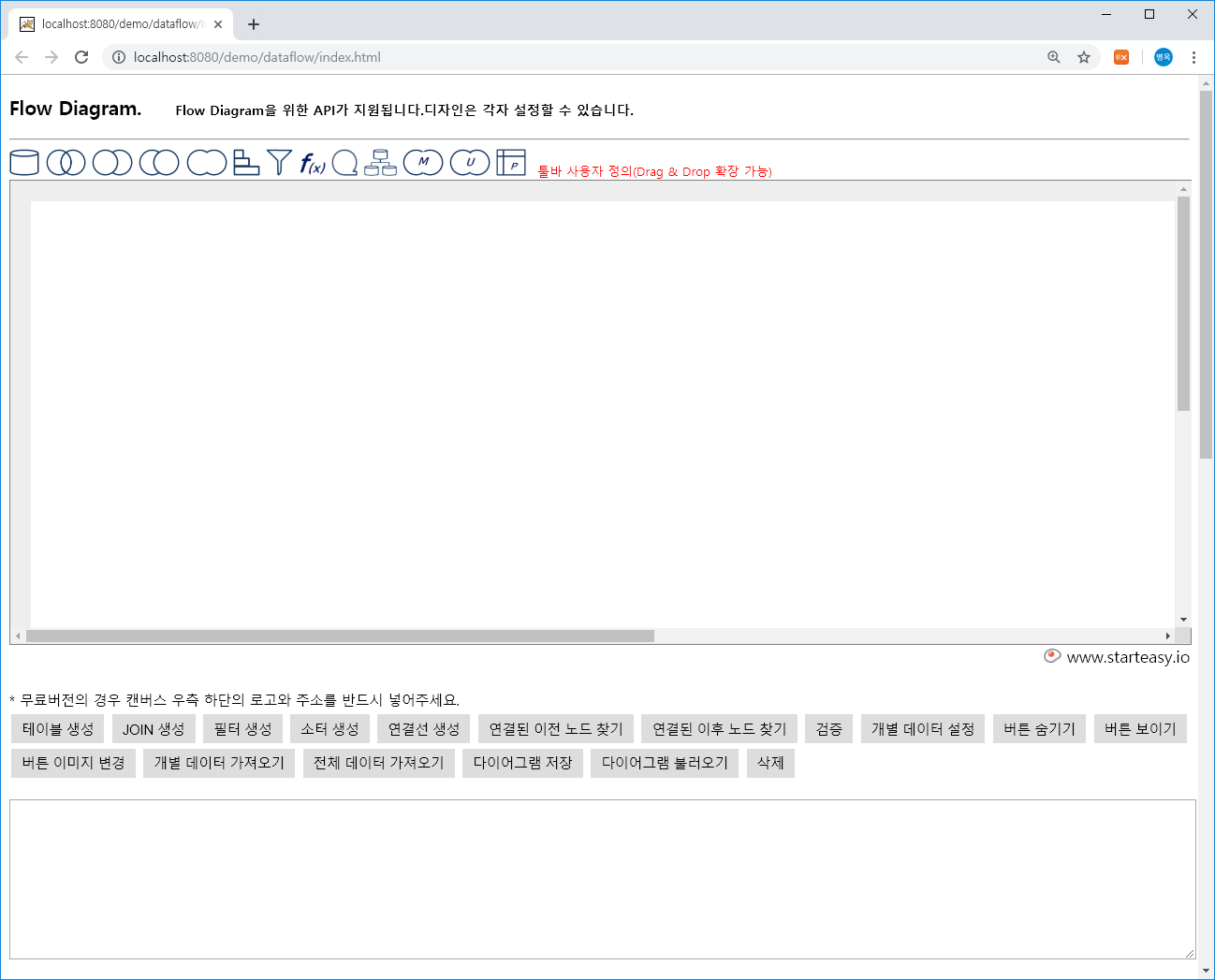
라이브러리 구조.



경로를 맞추고 브라우저에서 실행하면 화면을 확인할 수 있다.

화면 샘플과 도구바

이해를 돕기 위해 아래와 같은 화면을 기본코드로 넣어 놓았다. 사용자의 요구에 맞게 화면을 재구성하여 개발한다.



상단에 도구바가 존재하고 가운데 다이어그램을 나타낼 캔버스 영역과 하단에 버튼들을 두어 테스트를 가능하도록 했다.

상단의 도구 바는 여러분이 만들어서 사용할 내용이다. 도구바는 드래그 & 드롭이 가능해야 하므로 다음과 같이 코드를 작성할 수 있다.

<img src="./img/join-full-32px.png" height="26" ondragstart="drag(event, 'join', 'full')" draggable="true">

<img src="./img/sort-32px.png" height="26" ondragstart="drag(event, 'sort')" draggable="true">

<img src="./img/filter-32px.png" height="26" ondragstart="drag(event, 'filter')" draggable="true">

<img src="./img/function-32px.png" height="26" ondragstart="drag(event, 'function')" draggable="true">

각 이미지에 drag와 draggable 속성이 추가되어 있다. ondragstart의 두 번째 인자는 생성하려는 노느 명이고 세 번째 인자는 옵션이다.

이제 index.js 파일을 살펴보자. index.js에서 라이브러리를 로딩하고 사용자가 지정한 함수 등을 구현해서 사용한다. 현재 스타트이지 플로우 다이어그램은 Single 페이지에서도 두개의 인스턴스를 띄울 수 있는데 그 객체를 sed1으로 설정했다.

var sed1 = null;

window.addEventListener('load', function(){

//sedLib.setPath('../js/se', '../img/se', function(){

load();

//});

});

// 로딩

var load = function(){

sed1 = new Sed();

sed1.setDiv('div[ref=svgArea]', false);

sed1.exeMode = 'dashboard';

sed1.editMode = false;

//sed1.editPopMode = true;

sed1.diagramType = 'flowdiagram';

var options = {

width:2000,

height:800,

margin:20,

backgroundColor:'white',

backgroundImage:null

}

sed1.initDiagram(options);

sed1.editType = 'move';

... (생략)...

}

윈도우가 초기화 된 뒤 sed1을 초기화 시킨다.

setDiv는 캔버스 영역을 지정한다.

options의 값은 캔버스의 폭과 높이를 나타낸다.

이제 다시 툴바의 Drag & Drop을 구현하는 코드로 돌아와 보자. index.js파일의 setAction() 함수를 살펴보자. drag시 index.html에서 파라미터를 넘기고 있는데 이것들을 전역 변수로 설정해 놓는다. (차라리 전역으로 설정하는 것이 쉽다.) drop이 될 경우 전역 변수를 읽어와 툴바의 종류와 옵션을 확인하여 필요한 노드를 생성할 수 있다.

// 드래그 & 드롭 정의

var dropKind = null;

var dropKindSub = null;

var allowDrop = function(ev) {

ev.preventDefault();

}

var drag = function(ev, kind, kindSub) {

dropKind = kind;

dropKindSub = kindSub;

// 테이블일 경우 테이블명, 기타 경우에 따라 kind, 또는 ev.target.innerHTML

ev.dataTransfer.setData("text", kind);

}

var drop = function(ev) {

ev.preventDefault();

var data = ev.dataTransfer.getData("text");

if(dropKind == 'table'){

var fg = createTable(sed1, ev.offsetX, ev.offsetY);

fg.setText(data);

// 서버에서 데이터를 읽어와 객체에 할당하세요.

fg.getUserData().set('tableName', data);

fg.getUserData().set('columns',[

{name:'column1', type:'char'},

{name:'column2', type:'char'},

{name:'column3', type:'char'},

{name:'column4', type:'char'}

]);

} else if(dropKind == 'join'){

var fg = createJoin(sed1, ev.offsetX, ev.offsetY);

fg.setText(dropKind + '-' + dropKindSub);

fg.setJoinType(dropKindSub);

} else if(dropKind == 'filter'){

var fg = createFilter(sed1, ev.offsetX, ev.offsetY);

fg.setText(data);

} else if(dropKind == 'sort'){

var fg = createSort(sed1, ev.offsetX, ev.offsetY);

fg.setText(data);

} else if(dropKind == 'filter'){

var fg = createFilter(sed1, ev.offsetX, ev.offsetY);

fg.setText(data);

} else if(dropKind == 'function'){

var fg = createFunction(sed1, ev.offsetX, ev.offsetY);

fg.setText(data);

} else if(dropKind == 'csv'){

var fg = createCsv(sed1, ev.offsetX, ev.offsetY);

fg.setText(data);

} else if(dropKind == 'data-profile'){

var fg = createDataProfile(sed1, ev.offsetX, ev.offsetY);

fg.setText(data);

} else if(dropKind == 'data-wrangler'){

var fg = createDataWrangler(sed1, ev.offsetX, ev.offsetY);

fg.setText(data);

} else if(dropKind == 'file'){

var fg = createFile(sed1, ev.offsetX, ev.offsetY);

fg.setText(data);

} else if(dropKind == 'group'){

var fg = createGroup(sed1, ev.offsetX, ev.offsetY);

fg.setText(data);

} else if(dropKind == 'linear-regression'){

var fg = createLinearRegression(sed1, ev.offsetX, ev.offsetY);

fg.setText(data);

} else if(dropKind == 'merge'){

var fg = createMerge(sed1, ev.offsetX, ev.offsetY);

fg.setText(data);

} else if(dropKind == 'pearson-correlation'){

var fg = createPearsonCorrelation(sed1, ev.offsetX, ev.offsetY);

fg.setText(data);

} else if(dropKind == 'pivot'){

var fg = createPivot(sed1, ev.offsetX, ev.offsetY);

fg.setText(data);

} else if(dropKind == 'union'){

var fg = createUnion(sed1, ev.offsetX, ev.offsetY);

fg.setText(data);

} else if(dropKind == 'workflow'){

var fg = createWorkflow(sed1, ev.offsetX, ev.offsetY);

fg.setText(data);

} else if(dropKind == 'workflow-file'){

var fg = createWorkflowFile(sed1, ev.offsetX, ev.offsetY);

fg.setText(data);

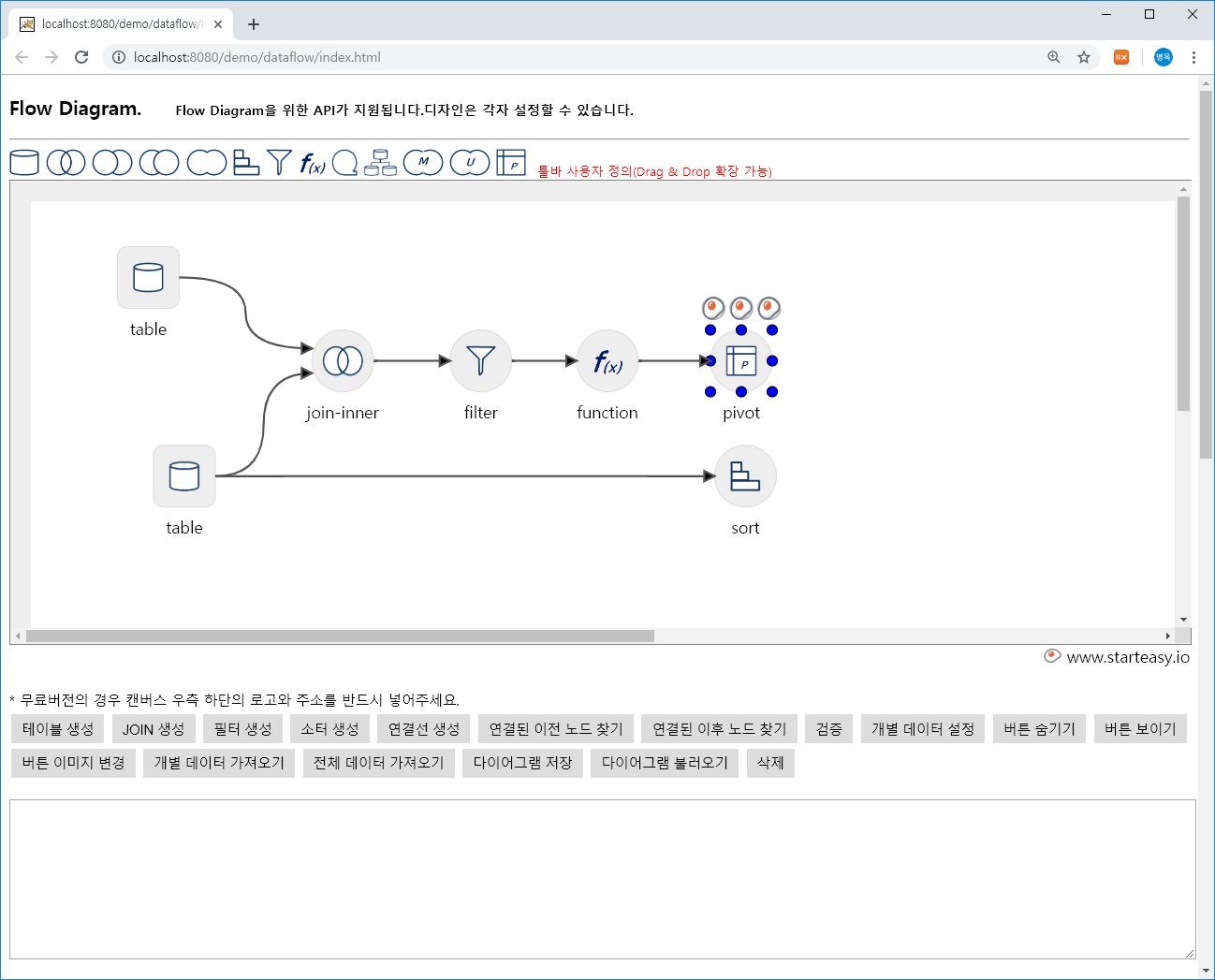
} else if(dropKind == 'flowline'){

var fg = createFlowline(sed1, ev.offsetX, ev.offsetY);

}

}

노드 정의하기



createTable은 테이블을 생성하는 함수이며 createTable은 define.js에 정의되어 있다.

createTable 함수를 살펴보자.

// 테이블 도형 정의

var createTable = function(pSed, x, y){

var options = {

figure: 'diagram', // 고정됨

shape: 'rect', // rect, ellipse 선택

width: 60, // rect의 가로크기

height: 60, // rect의 가로크기

rx : 10, // ellipse의 가로반지름

ry : 10, // ellipse의 세로반지름

fill: '#eee', // 채우기색상

strokeWidth: 1, // 선의 굵기

stroke: '#ddd', // 선색

resizable: false, // 크기조정가능여부

rotate: 0, // 회전각도

rotatable: false, // 회전여부 true, false

textHAlign : 'center', // 텍스트의 가로정렬

textVAlign : 'beneath', // 텍스트의 세로정렬 beneath만 지원함.

imageSrc: './img/table-32px.png', // 배경이미지

imageWidth: 30, // 이미지의 가로폭

imageHeight: 30, // 이미지의 세로폭

btnShape: 'circle', // 버튼의 모양 circle만 지원함

btnArrange: 'top', // 버튼의 위치 top만 지원함

btnStrockWidth: 1, // 버튼의 선 굵기

btnStroke: 'darkgray', // 버튼의 선 색

btnFill: 'lightgray',// 버튼의 배경색

btnCount: 3, // 버튼의 개수

btnXGab: 5, // 버튼간 공백 폭

btnYGab: 10, // 버튼과 주 도형과의 세로 폭

btnImages: ['./img/info-24px.png', './img/grid-24px.png', './img/start-24px.png'], // 버튼들의 이미지

btnVisibles: ['visible', 'visible', 'visible'], // 버튼의 초기 보이기 설정

btnRadius: [11, 11, 11], // 버튼의 반지름

btnImageWidth: [22, 22, 22], // 버튼의 이미지 폭

btnImageHeight: [22, 22, 22], // 버튼의 이미지 높이

diagramType: 'table', // 다이어그램 타입(사용자지정: connectInputEnables에서 지정한 tpype만 연결됨)

connectInputEnables: [], // 입력으로 받을 다이어그램 타입

connectMode: ['direct', 'move'], // direct는 default, move는 도형의 중앙이 겹칠 경우 자동으로 연결선 생성

joinDotsPosData: [{ x: 50, y:0, dir:'E', inout:'out', limit:999, min:1}] // JOIN 점의 위치와 방향, 제한수 설정(여러개 가능)

};

var fg = createDiagramFg( pSed, x, y, options);

// 버튼에 대한 정의는 자신의 객체에 한하여 정의한다.

fg.group.buttons[0].addEventListener('click', function(){

var e = new CustomEvent('sed', {detail:{fg:fg, sedEvent:'btnclick', btnIndex:0}});

fg.sed.div.dispatchEvent(e);

});

fg.group.buttons[1].addEventListener('click', function(){

var e = new CustomEvent('sed', {detail:{fg:fg, sedEvent:'btnclick', btnIndex:1}});

fg.sed.div.dispatchEvent(e);

});

fg.group.buttons[2].addEventListener('click', function(){

var e = new CustomEvent('sed', {detail:{fg:fg, sedEvent:'btnclick', btnIndex:2}});

fg.sed.div.dispatchEvent(e);

});

fg.group.tag.addEventListener('click', function(event){

console.log('::::', event.buttons);

var e = new CustomEvent('sed', {detail:{fg:fg, sedEvent:'click'}});

fg.sed.div.dispatchEvent(e);

});

fg.group.img.addEventListener('click', function(event){

var e = new CustomEvent('sed', {detail:{fg:fg, sedEvent:'click'}});

fg.sed.div.dispatchEvent(e);

});

fg.group.tag.addEventListener('contextmenu', function(event){

console.log('::::', event.buttons);

var e = new CustomEvent('sed', {detail:{fg:fg, sedEvent:'contextmenu'}});

fg.sed.div.dispatchEvent(e);

});

fg.group.img.addEventListener('contextmenu', function(event){

var e = new CustomEvent('sed', {detail:{fg:fg, sedEvent:'contextmenu'}});

fg.sed.div.dispatchEvent(e);

});

return fg;

}

shape: 모양을 나타낸다. rect일 경우 사각형, ellipse을 경우 원이나 타원으로 모양설정이 가능하다.

width: 노드의 폭을 나타낸다.

height: 노드의 높이를 나타낸다.

rx, ry : 사각형 모양일 경우 모서리의 라운드 크기를 나타낸다.

fill: 채우기 색상을 나타낸다.

strokeWidth: 선의 폭을 나타낸다.

stroke: 선의 색을 나타낸다.

resizable: 크기를 변경할 수 있는지 없는지를 설정한다.

rotate: 회전각도를 나타낸다.

rotatable: 회전가능여부를 나타낸다.

textHAlign: 텍스트의 가로정렬을 나타낸다. ‘center, left, right’.

textVAlign: 텍스트의 세로정렬을 나타낸다. ‘top, middle, bottom, beneath’.

imageSrc: 노드 가운데 이미지 경로를 설정한다.

imageWidth: 이미지의 폭을 설정한다.

imageHeight: 이미지의 높이를 설정한다.

btnShape: 노드 주변의 버튼모양을 설정한다. ‘circle’만 지원함.

btnArrange: 노드 주변의 버튼 위치를 설정한다. ‘top’만 지원함.

btnStrokeWidth : 노드 주변의 버튼 선의 폭을 설정한다.

btnStroke: 노드 주변의 버튼 선의 색을 설정한다.

btnFill: 노드 주변의 버튼 채우기 색을 설정한다.

btnCount: 노드 주변의 버튼의 개수를 설정한다.

btnXGab: 버튼간의 가로 여백을 설정한다.

btnYGab: 노드와 주변 버튼간의 세로 여백을 설정한다.

btnImages: 노드 버튼의 개수만큼 이미지 경로를 설정한다.

btnVisibles: 노드 주변 버튼의 초기 보이기를 설정한다.

btnImageWidth: 노드 주변 버튼의 이미지 폭을 설정한다.

btnImageHeight: 노드 주변 버튼의 이미지 높이를 설정한다.

**diagramType: 다이어그램의 종류를 나타낸다. (이것으로 연결여부 등을 결정한다. 사용자가 지정한다.)**

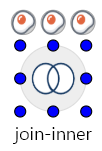
**connectEnables: 연결가능한 소스 노드를 배열로 설정한다. [‘table’, ‘join’] 이외의 노드는 연결할 수 없다.**

connectMode: 연결선을 노드에 직접연결 할지 다른 노드와 겹치면 자동으로 생길지를 결정한다. direct와 move를 보통 같이 설정한다.

joinDotPosData: 연결점을 설정하고 속성을 설정한다. 배열이므로 연결점을 여러 개 설정할 수 있다. 폭을 100% 기준으로 놓고 숫자를 생각한다. 왼쪽이면 -50, 오른쪽이면 50, 위면 -50, 아래면 50으로 생각한다.

[{ x: 50, y:0, dir:'E', inout:'out', limit:999, min:1}]

위 코드는 가로 오른쪽 끝 세로로 중앙인 점을 나타낸다. dir는 연결선의 방향인데 동쪽으로 연결선이 나가야 하기 때문에 ‘E’ (east)로 설정했다. inout은 out인데 이것은 연결선의 시작점이 화살표가 없는 쪽이 된다. in일 경우 화살표가 있는 방향의 점하고만 연결된다. limit:999는 999개까지 연결할 수 있다. 만약 하나면 연결하려 한다면 limit:1로 설정한다. 이경우 하나의 연결선이 연결되면 다른 연결선은 연결되지 않는다. min:1은 최소 1개의 연결선이 연결되어야 하며 연결되지 않을 경우 검증할 때 체크된다.



노드의 이벤트 관리

모든 이벤트는 sed 이벤트란 이름으로 발생한다. 한곳에서 모든 이벤트를 받아낼 수 있다.

// 상단버튼

fg.group.buttons[0].addEventListener('click', function(){

var e = new CustomEvent('sed', {detail:{fg:fg, sedEvent:'btnclick', btnIndex:0}});

fg.sed.div.dispatchEvent(e);

});

fg.group.buttons[1].addEventListener('click', function(){

var e = new CustomEvent('sed', {detail:{fg:fg, sedEvent:'btnclick', btnIndex:1}});

fg.sed.div.dispatchEvent(e);

});

fg.group.buttons[2].addEventListener('click', function(){

var e = new CustomEvent('sed', {detail:{fg:fg, sedEvent:'btnclick', btnIndex:2}});

fg.sed.div.dispatchEvent(e);

});

// 가운데 노드 클릭

fg.group.tag.addEventListener('click', function(event){

console.log('::::', event.buttons);

var e = new CustomEvent('sed', {detail:{fg:fg, sedEvent:'click'}});

fg.sed.div.dispatchEvent(e);

});

// 가운데 이미지 클릭( 위 노드와 동일하게 취급)

fg.group.img.addEventListener('click', function(event){

var e = new CustomEvent('sed', {detail:{fg:fg, sedEvent:'click'}});

fg.sed.div.dispatchEvent(e);

});

// 마우스 오른쪽 버튼

fg.group.tag.addEventListener('contextmenu', function(event){

console.log('::::', event.buttons);

var e = new CustomEvent('sed', {detail:{fg:fg, sedEvent:'contextmenu'}});

fg.sed.div.dispatchEvent(e);

});

fg.group.img.addEventListener('contextmenu', function(event){

var e = new CustomEvent('sed', {detail:{fg:fg, sedEvent:'contextmenu'}});

fg.sed.div.dispatchEvent(e);

});

index.js 에서 날아온 이벤트를 한곳에서 캐치해 낸다. 객체와 diagramType, 이벤트 종류를 알 수 있으므로 경우에 따른 설정이 가능하다.

sed1.div.addEventListener('sed', function(e){

console.log('arguments', e.detail.fg, e.detail.sedEvent, e.detail.btnIndex);

// 모든 제어는 여기에서...

var fg = e.detail.fg;

console.log(fg.options.diagramType)

if(fg.options.diagramType == 'table'){

if(e.detail.sedEvent == 'btnclick'){ // 이벤트를 용도에 맞게 정의할 것.

//alert('테이블의 버튼을 클릭함');

/\*

fg.setFill('yellow');

fg.setStroke('red');

fg.setStrokeWidth(3);

fg.hideButton(1); // index

fg.setText('<div color="blue" style="text-align:center">change table</div>');

fg.setTextVAlign('beneath');//bottom, top, middle

fg.setImage();

fg.setHidden();// 숨기기

fg.changeButtonImage(index, url) // 버튼 이미지 수정.

\*/

fg.setFontSize(28); // text가 div로 감싸 있을 경우 적용되지 않음

fg.setFontColor('orange'); // text가 div로 감싸 있을 경우 적용되지 않음

}

}

}, false)

**그 밖에 API 지원함수**

코드로 테이블 생성.

createTable(sed1, 100, 100);

감추기

sed1.svg.selectedFg.setHidden(‘hidden’);

보이기

sed1.svg.selectedFg.setHidden(‘visible’);

높이설정

sed1.svg.selectedFg.setHeight(100);

폭 설정

sed1.svg.selectedFg.setWidth(100);

선 굵기 설정

sed1.svg.selectedFg.setStrokeWidth(10);

선색 설정

sed1.svg.selectedFg.setStroke (‘red’);

채우기색 설정

sed1.svg.selectedFg.setFill (‘red’);

폰트크기 설정하기

sed1.svg.selectedFg.setFontSize(16);

텍스트 설정하기

sed1.svg.selectedFg.setText(‘AAA’);

텍스트 색상 설정하기

sed1.svg.selectedFg.setFontColor(‘red’);

코드로 좌표 이동하기

sed1.svg.selectedFg.moveXY(100, 100);

0번째 버튼 감추기

sed1.svg.selectedFg.hideButton(0);

0번째 버튼 보이기

sed1.svg.selectedFg.showButton(0);

이미지 변경하기

sed1.svg.selectedFg.setImage('./img/test.png');

버튼 이미지 변경하기 - 첫번째 인자는 버튼의 index 임.

sed1.svg.selectedFg.changeButtonImage(0, './img/arrow\_up.png');

고유번호로 노드찾기

sed1.findCompByFigureId(고유번호);

현재 노드의 연결점에서 In 노드와 Out노드 찾기

getDiagramInOutByIndex(fg, 0);

현재 노드의 In 노드와 Out노드 찾기

getDiagramInOut (fg);

이전노드 찾기

getPrevNode (fg);

이후노드 찾기

getNextNode (fg);

검증 모듈

var ret = verifyDiagram(sed1);

다이어그램 저장하기 데이터

sed1.getObjects();

다이어그램 불러오기

sed1.loadObjects(document.getElementById('dataView').value);

노드에 데이터에 속성을 설정한다.

sed1.svg.selectedFg.getUserData().set('item1', '푸르름');

노드에 데이터에 속성을 불러온다.

sed1.svg.selectedFg.getUserData().get('item1’);

노드의 데이터를 가져온다.

sed1.svg.selectedFg.getUserData()

전체 다이어그램의 노드연계 데이터 가져오기 (다이어그램 데이터는 다이어그램을 나타내는 데이터이고 노드연계 데이터는 연결정보, 테이블 정보 등 로직 구현에 필요한 데이터다.)

var jsonData = getDiagramDataAll(sed1);

노드 삭제하기

sed1.removeFg(sed1.svg.selectedFg);