addLink: function(link, override) {  
 if (!link.source || !link.target) {console.error('Illegal Link');}  
 link.layoutId = link.layoutId || ((link.klass || '') + ':' + (link.id || link.name || ''));  
 link.layoutType = 'link';  
 if (this.linkData.indexed[link.layoutId]) {  
 if (override) {  
 this.updateElement(link);}  
 if(link.klass === 'entity:entity')  
 {  
 // update the list of graphs in the link  
 this.linkData.indexed[link.layoutId] = link;  
 var linksIndex = $.each(this.links, function( eId, e){  
  
 if(e.hasOwnProperty('layoutId'))  
 {  
 if(e['layoutId'] === link.layoutId)  
 return eId;  
 }  
 } );  
 this.links[linksIndex] = link;  
 var foundValue, found = false;  
 var A = [];  
 A = this.linkData[link.klass];  
 var linkDataID = $.each(A, function( eId, e){  
  
 if(e.hasOwnProperty('layoutId'))  
 {  
 if(e['layoutId'] === link.layoutId)  
 return eId;  
 }  
 } );  
  
 A[linkDataID] = link;   
  
 }  
 return null;}  
 this.links.push(link);  
 this.linkData.indexed[link.layoutId] = link;  
  
 if (link.klass) {  
 this.linkData[link.klass] = this.linkData[link.klass] || [];  
 this.linkData[link.klass].push(link);  
 }  
 if (link.source && link.source.layoutId) {  
 // Make sure we're not duplicating the node  
 if (this.nodeData.indexed[link.source.layoutId]) {  
 link.source = this.getNode(link.source.layoutId);  
 }  
 // Add this link to the node.  
 this.applyToNode(link.source.layoutId, function(node) {node.links.push(link);});  
 }  
 if (link.target && link.target.layoutId) {  
 if (this.nodeData.indexed[link.target.layoutId]) {  
 link.target = this.getNode(link.target.layoutId);}  
 this.applyToNode(link.target.layoutId, function(node) {node.links.push(link);});}  
 if (this.linkAddTriggers[link.layoutId]) {  
 this.linkAddTriggers[link.layoutId].forEach(function(callback) {callback(link);});  
 delete this.linkAddTriggers[link.layoutId];}  
 return true;  
 },

"nodes": [ { "id": 1, "location": 'A', "type": 'protein' },  
 { "id": 2, "location": 'B', "type": 'protein' },  
 { "id": 3, "location": 'C', "type": 'protein' },  
 { "id": 4, "location": 'D', "type": 'protein' },  
 { "id": 5, "location": 'A', "type": 'protein' },  
 { "id": 6, "location": 'B', "type": 'protein' },  
 { "id": 7, "location": 'C', "type": 'protein' },  
 { "id": 8, "location": 'D', "type": 'protein' },  
 { "id": 9, "location": 'A', "type": 'protein' },  
 { "id": 10, "location": 'B', "type": 'protein' },  
 { "id": 11, "location": 'C', "type": 'protein'},  
 { "id": 12, "location": 'D', "type": 'protein' },  
 { "id": 13, "location": 'A', "type": 'protein' },  
 { "id": 14, "location": 'B', "type": 'protein' },  
 { "id": 15, "location": 'C', "type": 'protein'},  
 { "id": 16, "location": 'D', "type": 'protein' },  
 { "id": 17, "location": 'A', "type": 'protein' },  
 { "id": 18, "location": 'B', "type": 'protein' }  
 ],  
  
"links": [ { "target": 11, "source": 0 },  
 { "target": 3, "source": 0 },  
 { "target": 10, "source": 0 },  
 { "target": 16, "source": 0 },  
 { "target": 3, "source": 0 },  
 { "target": 9, "source": 0 },  
 { "target": 5, "source": 0 },  
 { "target": 11, "source": 0 },  
 { "target": 13, "source": 0 },  
 { "target": 16, "source": 0 },  
 { "target": 3, "source": 1 },  
 { "target": 9, "source": 1 },  
 { "target": 12, "source": 1 },  
 { "target": 4, "source": 2 },  
 { "target": 6, "source": 2 },  
 { "target": 8, "source": 2 },  
 { "target": 13, "source": 2 },  
 { "target": 10, "source": 3 },  
 { "target": 16, "source": 3 },  
 { "target": 9, "source": 3 },  
 { "target": 7, "source": 3 },  
 { "target": 11, "source": 5 },  
 { "target": 13, "source": 5 },  
 { "target": 12, "source": 5 },  
 { "target": 8, "source": 6 },  
 { "target": 13, "source": 6 },  
 { "target": 10, "source": 7 },  
 { "target": 11, "source": 7 },  
 { "target": 17, "source": 8 },  
 { "target": 13, "source": 8 },  
 { "target": 11, "source": 10 },  
 { "target": 16, "source": 10 },  
 { "target": 13, "source": 11 },  
 { "target": 14, "source": 12 },  
 { "target": 14, "source": 12 },  
 { "target": 14, "source": 12 },  
 { "target": 15, "source": 12 },  
 { "target": 16, "source": 12 },  
 { "target": 15, "source": 14 },  
 { "target": 16, "source": 14 },  
 { "target": 15, "source": 14 },  
 { "target": 16, "source": 15 },  
 { "target": 16, "source": 15 },  
 { "target": 17, "source": 16 }  
 ], \*/

/\*  
var force = d3.layout.force()  
 .size([config.w, config.h])  
 .nodes(nodes)  
 .links(links);  
  
force.linkDistance(config.w/3.5);  
var link = self.svg.selectAll('.link')  
 .data(links)  
 .enter().append('line')  
 .attr('class', 'link')  
 .attr('x1', function(d) {return nodes[d.source].x; } )  
 .attr('y1', function(d) {return nodes[d.source].y; } )  
 .attr('x2', function(d) {return nodes[d.target].x; } )  
 .attr('y2', function(d) {return nodes[d.target].y; } )  
 ;  
  
var node = self.svg.selectAll('.node')  
 .data(nodes)  
 .enter().append('circle')  
 .attr('class', 'node')  
 .attr('r', config.w/100)  
 .attr('cx', function(d) {return d.x;})  
 .attr('cy', function(d) {return d.y;});  
  
var animating = false;  
var animationStep = 400;  
force.on('tick', function(){  
 node.transition().ease('linear').duration(animationStep)  
 .attr('cx', function(d) {return d.x;} )  
 .attr('cy', function(d) {return d.y;} );  
  
 link.transition().ease('linear').duration(animationStep)  
 .attr('x1', function(d) {return d.source.x;})  
 .attr('y1', function(d) {return d.source.y;})  
 .attr('x2', function(d) {return d.target.x;})  
 .attr('y2', function(d) {return d.target.y;});  
  
 force.stop();  
  
 if(animating)  
 {  
 setTimeout( function() {force.start();}, animationStep );  
 }  
  
}); \*/  
  
//self.layout.force.start();  
  
/\*  
force.on('end', function(){  
 node.attr('r', config.w/100)  
 .attr('cx', function(d) {return d.x;})  
 .attr('cy', function(d) {return d.y;});  
  
 link.attr('x1', function(d) {return d.source.x;})  
 .attr('y1', function(d) {return d.source.y;})  
 .attr('x2', function(d) {return d.target.x;})  
 .attr('y2', function(d) {return d.target.y;});  
 });  
\*/  
//animating = true;  
//force.start();  
  
  
//root.append('<div id="drag" style="width: 50px; height: 50px; background-color: red;"/>');  
//root.append('TEST');  
//root.append('<hr/>');  
  
/\*  
root.find('#drag').draggable({  
 revert: true,  
 revertDuration: 0,  
 scroll: false,  
 stop: function(event, ui) {  
 var force;  
  
 if (self.contains(mouse.x, mouse.y)) {return;}  
  
 var mouse = $P.state.mainCanvas.getMouseLocation(event);  
 mouse.x += $P.state.scrollX;  
  
 var send = {  
 name: 'dragPathway',  
 x: mouse.x, y: mouse.y,  
 strokeStyle: 'gray',  
 expression: null};  
 var result = $P.state.scene.sendEvent(send);  
  
 if (!result) {  
 force = new $P.Bubble.Force({x: mouse.x, y: mouse.y, w: 750, h: 600});  
 $P.state.scene.add(force);  
 result = force.receiveEvent(event);}  
  
 if (result && result.addLink) {  
 // Add Link Here.  
 }  
  
}});  
  
root.find('#search\_run').click(function(event) {  
 self.updateSearch();});  
\*/