You might use [lineTo](https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D/lineTo) from canvas context. See <https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D> for more details.

First, you define a canvas on html:

<canvas id="connection-canvas"></canvas>

Then you might draw a line on it:

function drawLine(p1, p2) {

var canvas = document.getElementById("connection-canvas");

var ctx = canvas.getContext("2d");

ctx.beginPath();

ctx.moveTo(p1.x, p1.y);

ctx.lineTo(p2.x, p2.y);

ctx.stroke();

}

All you need is to figure out the offset of your "round-pointers":

function getPoint(answerElement) {

var roundPointer = answerElement.lastElementChild;

return {

y: answerElement.offsetTop + roundPointer.offsetTop + roundPointer.offsetHeight / 2,

x: answerElement.offsetLeft + roundPointer.offsetLeft + roundPointer.offsetWidth / 2

};

}

So you could with these two functions get the center point of a "round-pointer" and draw a line in canvas from it to another "round-pointer", assuming that the canvas is positioned on same answer-container's offset parent and its size must be large enough.

Next you have to deal with selecting the two answers to connect them. Below there is a demo for you. This demo does not handle erasing a line, if you change an answer.

var lastSelection;

// Add click listener for answer-container

function listenToClick() {

var rows = document.querySelectorAll('.row'),

row;

var cols, col;

for (row = 0; row < rows.length; row++) {

cols = rows[row].children;

for (col = 0; col < cols.length; col++) {

// Bind information about which answer is this,

// so we can prevent from connecting two answers on

// same column.

cols[col].addEventListener('click', selectAnswer.bind({

row: row,

col: col,

element: cols[col]

}));

}

}

}

// This is fired when a answer-container is clicked.

function selectAnswer(event) {

if (lastSelection) {

lastSelection.element.classList.remove('selected');

}

if (!lastSelection || lastSelection.col === this.col) {

lastSelection = this;

this.element.classList.add('selected');

} else {

drawLine(getPoint(this.element), getPoint(lastSelection.element));

lastSelection = null;

}

}

function getPoint(answerElement) {

var roundPointer = answerElement.lastElementChild;

return {

y: answerElement.offsetTop + roundPointer.offsetTop + roundPointer.offsetHeight / 2,

x: answerElement.offsetLeft + roundPointer.offsetLeft + roundPointer.offsetWidth / 2

};

}

function drawLine(p1, p2) {

var canvas = document.getElementById("connection-canvas");

var ctx = canvas.getContext("2d");

ctx.beginPath();

ctx.moveTo(p1.x, p1.y);

ctx.lineTo(p2.x, p2.y);

ctx.stroke();

}

function resizeCanvas() {

var canvas = document.getElementById("connection-canvas");

var ctx = canvas.getContext("2d");

ctx.canvas.width = window.innerWidth;

ctx.canvas.height = window.innerHeight;

}

listenToClick();

resizeCanvas();

.padding-answer-line-mapping

{

padding-bottom:8px;

}

.answer-container

{

width:40px;

height:40px;

background-color:#ccc;

border:1px solid #ccc;

margin:2px;

float:left;

text-align:center;

padding-top:8px;

cursor:pointer;

position:relative;

}

.answer-container:hover, .answer-container:focus, .answer-container:active

{

background-color: #0076e9;

color: white;

border: 1px solid #0076e9;

}

.round-pointer-right

{

position: absolute;

background-color:#ccc;

cursor:pointer;

width:10px;

height:10px;

border-radius: 50%;

right:0px;

top:14px;

margin-right:-20px;

}

.round-pointer-left

{

position: absolute;

background-color:#ccc;

cursor:pointer;

width:10px;

height:10px;

border-radius: 50%;

left:0px;

top:14px;

margin-left:-20px;

}

.selected {

background-color: red;

}

<link href="//code.ionicframework.com/1.3.1/css/ionic.css" rel="stylesheet"/>

Match the following items.

<canvas id="connection-canvas" style="position: absolute; top: 0; left: 0; right: 0; bottom: 0"></canvas>

<div class="row padding-answer-line-mapping">

<div class="col answer-container">

One

<div class="round-pointer-right"></div>

</div>

<div class="col">

</div>

<div class="col answer-container">

2

<div class="round-pointer-left"></div>

</div>

</div>

<div class="row padding-answer-line-mapping">

<div class="col answer-container">

Two

<div class="round-pointer-right"></div>

</div>

<div class="col">

</div>

<div class="col answer-container">

1

<div class="round-pointer-left"></div>

</div>

</div>

<mat-form-field class="example-form-field">

<input matInput type="text" placeholder="Clearable input" [(ngModel)]="value">

<button mat-button \*ngIf="value" matSuffix mat-icon-button aria-label="Clear" (click)="value=''">

<mat-icon>close</mat-icon>

</button>

</mat-form-field>

import {Component} from '@angular/core';

*/\*\**

*\* @title Input with a clear button*

*\*/*

@Component({

selector: 'input-clearable-example',

templateUrl: './input-clearable-example.html',

styleUrls: ['./input-clearable-example.css'],

})

export class InputClearableExample {

value = 'Clear me';

}

.example-form-field {

width: 200px;

}