Abstract.

1. Index.js

Here, receives all inputs from the interface(HTML) and sends parameters to each function that draws responsible templates.

And makes all interfaces of objects defined in object.js with id(defined in HTML)

1. Object.js(in the object Folder)

Defines the necessary objects such as Pathine, Rect, Circle …

Each objects has their methods.

     function Pathline(id) {

        this.drawPath = function (position) {

            $('#'+id).attr("d",'M ' + position[0].x + ' ' + position[0].y + ' L ' + position[1].x + ' '+ position[1].y)

        }

        this.getPosition = function() {

          this.path\_pos = [{x:0,y:0},{x:0,y:0}];

          this.position = $('#'+id).attr("d").split(" ")

          this.path\_pos[0].x = Number(this.position[1])

          this.path\_pos[0].y = Number(this.position[2]);

          this.path\_pos[1].x = Number(this.position[4]);

          this.path\_pos[1].y = Number(this.position[5]);

           return this.path\_pos;

        }

      }

You can make a instance with id defined in HTML in the index.js.

ex) Upline = new Pathline(“Uline”)

 function definitionObject() {

      //  = = == =  = = == = Main = = = = = == == = = ==

      //Line

      this.upLine = new Pathline("ULine");

      this.leftLine = new Pathline("LLine");

1. drawer Folder

Defines each classes responsible for drawing corresponding templates.

In each draw function, they handle all instances of objects that defined in index.js file.

Such as draw , set the size, get the present position so on.

Ex) circle.

      function circle(id) {

        this.moveCircle = function(position) {

          $('#'+id).attr({'cx':''+ position.x+'','cy':''+position.y+''});

        }

        this.setSize = function(size) {

          $('#'+id).attr({'rx':''+ size+'','ry':''+size+''});

        }

        this.getPosition = function() {

          this.circle\_pos = {x:0,y:0};

           this.positionX = $('#'+id).attr('cx')

          this.positionY = $('#'+id).attr('cy')

          this.circle\_pos.x = Number(this.positionX)

          this.circle\_pos.y = Number(this.positionY);

            return this.circle\_pos;

        }

      }

1. Function Folder

Wraps the additional functions necessary for drawing the templates.

Note.

When you input the value with fraction such as Height = 13-14/17 ,

Please be careful to input this ‘ - ’ between integer and fraction.

Example for each templates

1. **simple panel**

inputs = width, height, centerNote

Height = 59-7/8 , Width = 29-3/4 , Center Note = All Edges Flat Polished

After pass the input, if you click the “transform” button, then it is implemented in the index.js.

Here, assigns the inputted values to the corresponding variables.

In our case this.inputWidth, this.inputHeight, this.centerNote.

   $("#transform").click( function() {

    // mainValue(simplePanel)

      this.inputWidth = inputing("width");

      this.inputHeight = inputing("height")

      this.centerNote = $("#note").val();

The variables are passed in the context variable that are used as parameter wrapper.

  var context =

    {

      Canvas: null,

      Parameters: [

         { ParameterName: 'simeplePanel',

         value: [{width:this.inputWidth,height:this.inputHeight},{centerNote:this.centerNote}],  Type: 'fraction', FriendlyNmae : 'inputWidth' },

After that, this.draw() method of Transform class is implemented with this context parameter.

 var simplePanle = new simplePanel();

            DrawingManager.register(simplePanle);

            var drawer1 = DrawingManager.findDrawer('simplePanel');

            drawer1.draw(context);

simplePanel class are defined in the simple.js (in the drawer Folder).

And this.draw() function of this class are implemented with passed context parameter.

1. **Single template**

**Ex) corner**

Inputs

Height = 59-7/8 , Width = 29-3/4 , Center Note = All Edges Flat Polished

Left Angle Corner Height = 12-1/16 , Width = 13-1/4

All the processes are same above.

Receives values from the HTML with id and assigns that to variables in the index.js

 // cornerValue

      this.LU\_inputC\_Width = inputing("LU\_c\_width1");

      this.LU\_inputC\_Height = inputing("LU\_c\_height1");

and these are passed in context parameter wrapper.

  var context =

    {

      Canvas: null,

      Parameters: [

         { ParameterName: 'simeplePanel',

         value: [{width:this.inputWidth,height:this.inputHeight},{centerNote:this.centerNote}],  Type: 'fraction', FriendlyNmae : 'inputWidth' },

         { ParameterName: 'corner',

         value: [

            {width:this.LU\_inputC\_Width, height:this.LU\_inputC\_Height},

            {width:this.RU\_inputC\_Width, height:this.RU\_inputC\_Height},

            {width:this.LB\_inputC\_Width, height:this.LB\_inputC\_Height},

            {width:this.RB\_inputC\_Width, height:this.RB\_inputC\_Height},],   Type: 'fraction', FriendlyNmae : 'corner' },

   var draw\_corner = new corner();

            DrawingManager.register(draw\_corner);

            var drawer2 = DrawingManager.findDrawer('corner');

            drawer2.draw(context);

all the process are same like above until Miter.

The additional thing to remember is you have to input value for simple panel whenever you draw the each templates.

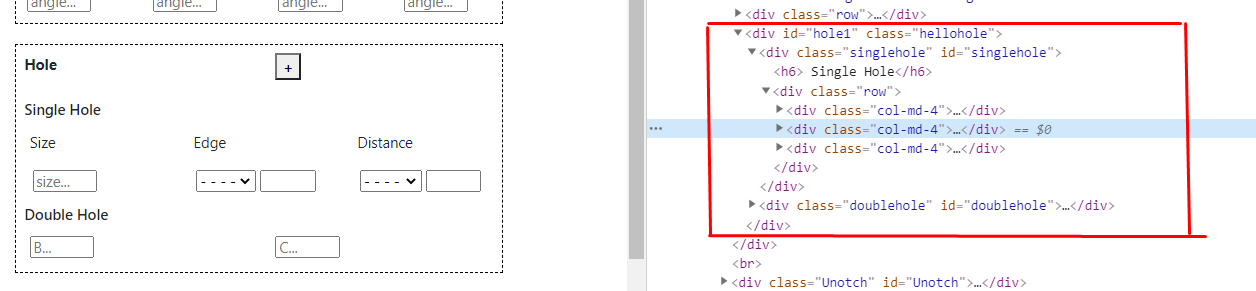
1. **Multiple template**

**Ex) Hole**

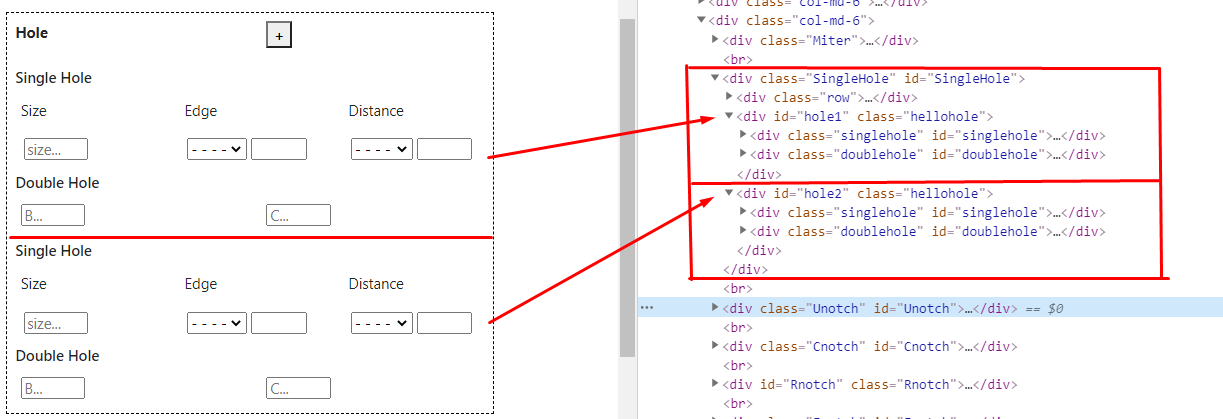
Until now, every elements for inputs in HTML are static.

But from now, they are all dynamic, i.e you can’t see input elements directly in the HTML file.

You can just see input elements for single and this is for interface at first.



When you clieck ‘+’ button for multiple, then elements are created dynamically in the HTML file.

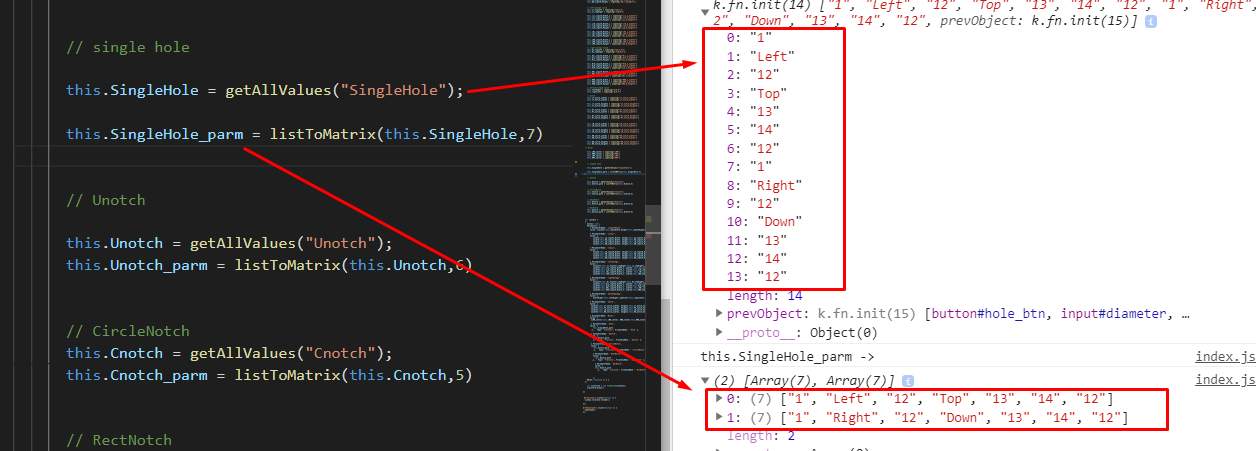


      // single hole

      this.SingleHole = getAllValues("SingleHole");

      this.SingleHole\_parm = listToMatrix(this.SingleHole,7)

if you input the “transform” button, then receives values from the HTML.



And the array of inputted values are passed in the context like above.

      { ParameterName: 'Hole',

         value: [

            this.SingleHole\_parm

          ,],   Type: 'fraction', FriendlyNmae : 'Hole' },

And draws the templates with this context parameter.

  var draw\_Hole = new Hole();

            DrawingManager.register(draw\_Hole);

            var drawer9 = DrawingManager.findDrawer('Hole');

            drawer9.draw(context);

\*\*\*\*

Detail explain for drawing.

 this.draw = function(context) {

          var simpleParameter = [];

          simpleParameter = context.Parameters.filter((parameter, i) => {

           return parameter.ParameterName === 'Hole';

         })

Finds the corresponding parameters from the context with the parameterName defined in context.

And assigns these values to member variable of class.

this.diameter = this.parameter[i][0]?strTofraction(this.parameter[i][0]):0;

  this.set\_direct = this.parameter[i][1];

  this.setback = this.parameter[i][2]?strTofraction(this.parameter[i][2]):0;

  this.dis\_setback =this.parameter[i][2]?strTofraction(this.parameter[i][2]):0;

and defines the instances of objects to draw with id created dynamically.

   var singleCircle = new circle('singleCircle'+i+'');

        var SU\_lArrowline = new Pathline('S\_LU\_Line'+i+'');

        var SU\_lfupArrowP = new Pathline('S\_LU\_P'+i+'');

        var SU\_lfdownArrowP = new Pathline('S\_LD\_P'+i+'');

        var SU\_lfupTip = new Arrow('S\_LU\_UTip'+i+'','up');

        var SU\_lfdownTip = new Arrow('S\_LU\_DTip'+i+'','down');

        var SH\_group = new Group('SH\_group'+i+'', 'vertical')

and handle these objects with inputted values using these object’s method.

 this.sing\_position = {x:ele.rightLine.getPosition()[0].x - this.setback\*10\*b.rate,y: ele.downLine.getPosition()[0].y - this.distance\*10\*b.rate};

 singleCircle.moveCircle(this.sing\_position)