Get numerical point data

Returns the x-coordinate of point name
Part::x(name);

Returns the y-coordinate of point name
Part::y(name);

Returns the x-axis delta between points name1 and name2
Part::deltaX(name1, name2);

Returns the x-axis delta between points name1 and name2
Part::deltaY(name1, name2);

Returns the angle made by a line from points name1 to name2
Part::angle(name1, name2);

Returns the length of curve {start,cp1,cp2,end}
Part::curveLen(start, cp1, cp2, end);

Returns the distance between points name1 and name2
Part::distance(name1, name2);

Various

Returns true if point name exists in the part
Part::isPoint(name);

Returns the part title
Part::getTitle( );

Sets the part render flag to bool
Part::setRender(bool);

Returns distance as a formatted string with the correct units
Part::unit(distance);

Generates a new unique id with optional prefix
Part::newId(prefix);

Path offset

Offset path source as new path name at offset, render and attributes are optional
Part::offsetPath(name, source, offset, render[false], attributes);

Offset pathstring as new path name at offset, render and attributes are optional
Part::offsetPathString(name, pathstring, offset, render[false], attributes);

Adding points based on lines/curves

Add point at intersection of line segments {fromA,toA} and {fromB,toB}
Part::linesCross(fromA, toA, fromB, toB);

Add point at intersection of lines {fromA,toA} and {fromB,toB}
Part::beamsCross(start1, end1, start2, end2);

Add point at edge edge of curve {start,cp1,cp2,end}
edge is one of: left,right,top,bottom
Part::curveEdge(start, cp1, cp2, end, edge);

Add points crossing curve {start,cp1,cp2,end} at x-coord, prefix is optional
Part::curveCrossesX(start, cp1, cp2, end, x-coord, prefix);

Add points crossing curve {start,cp1,cp2,end} at y-coord, prefix is optional
Part::curveCrossesY(start, cp1, cp2, end, x-coord, prefix);

Add points at intersections between curve {start,cp1,cp2,end} and line {from,to}, prefix is optional
Part::curveCrossesLine(start, cp1, cp2, end, from, to, prefix);

Add points at intersections between curves {startA,cp1A,cp2A,endA} and {startB,cp1B,cp2B,endB}, prefix is optional
Part::curvesCross(startA, cp1A, cp2A, endA, startB, cp1B, cp2B, endB, prefix);

Add points to split curve {start,cp1,cp2,end} in two halves at split, prefix and splitOnDelta are optional
If splitOnDelta is true, split must be a value between 0 and 1. If not, it's the name of the point to split on.
Part::splitCurve(nameStart, nameCp1, nameCp2, nameEnd, nameSplit, prefix , splitOnDelta[false]);

Adding points

Adds point as name, description is optional
Part::addPoint(name, point, description);

Adds point name with coordinates x-coord and y-coord, description is optional
Part::newPoint(name, x-coord, y-coord, description);

Clones point source into point name
Part::clonePoint(source, name);

Adding non-points

Adds message as text name anchored on anchor, attributes are optional
Part::newText(name, anchor, message, attributes);

Adds pathstring as path name, attributes is optional
Part::newPath(name, patstring, attributes);

Adds message as textOnPath name along pathstring, attributes are optional
Part::newTextOnPath(name, pathstring, message, attributes);

Adds message as note name anchored on anchor, hour, length, offset, and attributes are optional
Part::newNote(name, anchor, message, hour, length, offset, attributes);

Adds snippet name with defs id reference anchored on anchor, attributes are optional
Part::newSnippet(name, reference, anchor, attributes);

Adds include name with svg code svg
Part::newInclude(name, svg);

Adds a grainline path between from and to, text is optional
Part::newGrainline(from, to, text);

Adds a cut-on-fold path between from and to, text and offset is optional
Part::newCutonfold(from, to, text, offset);

Places a notch at each point in array points
Part::notch(points);

Adds title with number, title, and message anchored on anchor in optional mode
Mode is one of: default, vertical, horizontal, small, vertical-small, or horizontal-small
Part::addTitle(anchor, number, title, message, mode);

Adding dimensions

All these methods take 3 extra optional parameters at the end:
pathAttributes, labelAttributes, and leaderAttributes

Adds a width dimension from from to to at y-coord, text is optional
Part::newWidthDimension(from, to, y-coord, text);

Adds a height dimension from from to to at x-coord, text is optional
Part::newHeightDimension(from, to, x-coord, text);

Adds a linear dimension from from to to at offset, text is optional
Part::newLinearDimension(from, to, offset, text);

Adds a curved dimension at offset from pathstring, text is optional
Part::newCurvedDimension(pathstring, offset, text);

Adds a small width dimension from from to to at y-coord, text is optional
Part::newWidthDimensionSm(from, to, y-coord, text);

Adds a small height dimension from from to to at x-coord, text is optional
Part::newHeightDimensionSm(from, to, x-coord, text);

Adds a small linear dimension from from to to at offset, text is optional
Part::newLinearDimensionSm(from, to, offset, text);

Pattern methods

Set option name to value
Pattern::setOption(name, value);

Returns option name
Pattern::getOption(name);

Returns option name - Alias of getOption
Pattern::o(name);

Set value name to value
Pattern::setValue(name, value);

Returns value name - Alias of getValue
Pattern::getValue(name);

Returns value name
Pattern::v(name);

Translate message
Pattern::t(message);

Convert value to correct units
Pattern::unit(value);

Clone points from part from into part into
Pattern::clonePoints(from, into);

Add a new part with name name
Pattern::newPart(name);

Add message to the pattern messages
Pattern::msg(message);

Add message to the pattern debug messages
Pattern::dbg(message);

Returns true if this is a paperless pattern
Pattern::isPaperless( );

Model methods

Returns measurement name
Model::getMeasurement(name);

Returns measurement name - alias for getMeasurement
Model::m(name);

Sets measurement name to value
Model::setMeasurement(name, value);

BezierToolbox methods

Returns distance to control point to mimic a circle with radius
Note that this methos is static, no BezierToolbox object needed
BezierToolbox::bezierCircle(radius);

Freesewing cheat sheet

Notation legend
Class::method(object, numeric, string, array, bool , object, numeric, string, array, bool[default]);
optional