JessieScript Reference (Version Juni 2010 v0.81)

1 Construct

Easy mathematical constructions can be created with the line

```
board.construct(...);
```

elements, seperated by semicolon, as one string. Blanks are irrelevant.

Possible elements are:

Example	Description
`A(1,1)`	point at (1,1) with name `A`
`BB(-2|0.5)`	point at (-2,0.5) with name `BB`
`]AB[`	straight line through points `A` and `B`
`[AB[`	ray through points `A` and `B`, stopping at `A`
`]A BB]`	ray through points `A` and `BB`, stopping at `BB`
`[AB]`	segment between `A` and `B`
`g=[AB]`	segment between `A` and `B` with name `g`
`k(A,4)`	circle with midpoint `A` and radius 4
`k(A,[BC])`	circle with midpoint `A`, whose radius is given by the (not necessarily existing) segment `[BC]`
`k(A,B)`	circle with midpoint `A`, through point `B`
`k1=k(A,3)`	circle with midpoint `A` with radius 3 with name `k1`
`P(g)`	glider `P` on the object `g`
`Q(k1,0,1)`	glider `Q` on the object `k1` at (0,1)
`g&k1`	intersection point(s) of the objects `g` and `k1`
`S=g&k1`	intersection point(s) of the `g` and `k1`. Multiple intersection points are named with S_1 and S_2, single ones with `S`.
`||(A,g)`	parallel line to `g` through point `A`
`|_(A,g)`	perpendicular line to `g` through point `A`
`<(A,B,C)`	angle, defined by the points `A`, `B`, `C`
`alpha=<(A,B,C)`	angle, defined by the points `A`, `B`, `C`, with name α Possible greek denominators are `alpha`, `beta`, `gamma`, `delta`, `epsilon`, `zeta`, `eta`, `theta`, `iota`, `kappa`, `lambda`, `mu`, `nu`, `xi`, `omicron`, `pi`, `rho`, `sigmaf`, `sigma`, `tau`, `upsilon`, `phi`, `chi`, `psi` and `omega`.
`1/2(A,B)`	midpoint between `A` and `B`
`3/4(A,B)`	point dividing the segment from `A` to `B` at ratio 3:7 , i.e. $\frac{3}{4}$ parts of the segment `[AB]` are between `A` and the constructed point Therefore, any ratio of natural numbers is possible.
`P[A,B,C,D]`	polygon through points `A`, `B`, `C`, `D` with name 'P'
`f:x^2+2*x`	functiongraph, $f : x \mapsto x^2 + 2 \cdot x$
`f:sin(x)`	functiongraph, $g : x \mapsto \sin(x)$
`#Hello world(0,3)`	text `Hello world` at (0,3)

Its possible for every element (except points, graphs and polygons) to provide a name directly by using

```
objname = ...
```

the properties can still be changed.

2 Fast modification of properties

For setting the three most important properties there is a fast possibility, all others have to be set afterwards by accessing the particular elements and calling the corresponding function.
These are

Property	Description
`invisible`	the object is invisible
`draft`	the object is drawn in draft mode
`nolabel`	the object does not have a label

These properties are set directly at declaring the objects by writing the respective key word (resp. key words, a combination is possible), seperated by a blank behind the construction command before the semicolon, i.e.

```
P(1,1) nolabel; Q(2,3) draft nolabel; [PQ] invisible;
```

3 Zugriff auf Elemente

Access to the elements after constructing them is possible by using:

element	description
`constr.points[i]`	take the i-th point or glider of the construction `constr`, also midpoints and dividing points are within this array
`constr.lines[i]`	take the i-th line, ray or segment of the construction `constr`, also parallel and perpendicular lines are within this array
`constr.circles[i]`	take the i-th cicle of the construction `constr`
`constr.intersections[i]`	take the i-th intersection point of the construction `constr`
`constr.angles[i]`	liefert take the i-th angle of the construction `constr`
`constr.functions[i]`	take the i-th function graph of the construction `constr`
`constr.texts[i]`	take the i-th text element of the construction `constr`
`constr.polygons[i]`	take the i-th polygon of the construction `constr`
`constr.A`	take the element with name `A` of the construction `constr`

4 Macros

Additionally it is possible to define macros. The key word is `Marco`, the parameters are, seperated by comma, provided within round brackets, the content between curly braces. Left of the equal sign any name can be given to the macro.
So the syntax is given by

```
macroName = Macro(param1, param2, param3, ...) { command1; command2; command3; ... };
```

After that, the macro can be called by

```
result = macroName(x1,x2,x3,...);
```

5 Example

An example shall demonstrate the practical implementation.

```
board = JXG.JSXGraph.initBoard('box', {originX: 50, originY: 300, unitX: 50,
                                       unitY: 50, axis:true});

cons1 = board.construct("A(1,1);BC(1,3);k(A,[BC]);X(2,4)");
cons2 = board.construct("J(7,4);l_2=[BC A]");

cons1.points[0].face{'>'}; // A
cons1.BC.strokeColor('black');
cons2.l_2.strokeWidth(4);
cons1.X.size(8);

cons3 = board.construct("test = Macro(D,E,F) { g=[DE] nolabel; k1=k(D,[EF]);};
                                ttt=test(A,X,J);");
cons3.ttt.g.strokeColor('red');
```