JessieScript Referenz (Version Juni 2010 v0.81)

1 Konstruieren

Einfache mathematische JSXGraph-Konstruktionen können mit dem Befehl

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

erzeugt werden. Dabei können verschiedene Elemente, durch Semikolon getrennt als String übergeben werden. Leerzeichen spielen keine Rolle.

Mögliche Elemente sind:

Beispiel	Beschreibung
`A(1,1)`	Punkt an der Stelle (1,1) mit dem Namen `A`
`BB(-2|0.5)`	Punkt an der Stelle (-2,0.5) mit dem Namen `BB`
`]AB[`	Gerade durch die Punkte `A` und `B`
`[AB[`	Halbgerade durch die Punkte `A` und `B`, über `B` hinaus
`]A BB]`	Halbgerade durch die Punkte A und BB, über A hinaus
`[AB]`	Strecke zwischen `A` und `B`
`g=[AB]`	Strecke zwischen `A` und `B` mit dem Namen `g`
`k(A,4)`	Kreis um `A` mit Radius 4
`k(A,[BC])`	Kreis um `A`, dessen Radius durch die Länge der (nicht notwendigerweise existierenden) Strecke `[BC]`
`k(A,B)`	Kreis um `A`, dessen Kreislinie durch den Punkt `B` geht
`k1=k(A,3)`	Kreis um `A` mit Radius 3 mit dem Namen `k1`
`P(g)`	Gleiter `P` auf dem Objekt `g`
`Q(k1,0,1)`	Gleiter `Q` auf dem Objekt `k1` mit den Koordinaten (0,1)
`g&k1`	Schnittpunkt(e) der Objekte `g` und `k1`
`S=g&k1`	Schnittpunkt(e) der Objekte `g` und `k1`. Mehrere Schnittpunkte werden mit S_1 und S_2 bezeichnet, einzelne mit `S`.
`||(A,g)`	Parallele zur Geraden `g` durch den Punkt `A`
`|_(A,g)`	Senkrechte zur Geraden `g` durch den Punkt A
`<(A,B,C)`	Winkel, definiert durch die Punkte `A`, `B`, `C`
`alpha=<(A,B,C)`	Winkel, definiert durch die Punkte `A`, `B`, `C`, mit dem Namen α Mögliche griechische Bezeichner sind `alpha`, `beta`, `gamma`, `delta`, `epsilon`, `zeta`, `eta`, `theta`, `iota`, `kappa`, `lambda`, `mu`, `nu`, `xi`, `omicron`, `pi`, `rho`, `sigmaf`, `sigma`, `tau`, `upsilon`, `phi`, `chi`, `psi` und `omega`.
`1/2(A,B)`	Mittelpunkt von `A` und `B`
`3/4(A,B)`	Punkt, der die Strecke von `A` nach `B` im Verhältnis 3:7 innen teilt, d.h. $\frac{3}{4}$ der Strecke `[AB]` liegen zwischen `A` und dem Teilpunkt Dabei ist jedes Verhältnis natürlicher Zahlen möglich.
`P[A,B,C,D]`	Polygon durch die Punkte `A`, `B`, `C`, `D` mit dem Namen 'P'
`f:x^2+2*x`	Funktionsgraph, $f : x \mapsto x^2 + 2 \cdot x$
`f:sin(x)`	Funktionsgraph, $g : x \mapsto \sin(x)$
`#Hallo Welt(0,3)`	Text `Hallo Welt` an den Koordinaten (0,3)

Es ist für jedes der Elemente (außer Punkte, Graphen und Polygone) möglich, mit

```
objname = ...
```

direkt einen Namen zu vergeben.
Die Funktion gibt ein Objekt mit allen erzeugten Elementen zurück, sodass danach noch Eigenschaften verändert werden können.

2 Schnelles Verändern von Eigenschaften

Zum Setzen der drei wichtigsten Eigenschaften gibt es eine schnelle Möglichkeit, alle anderen müssen im Nachhinein durch Zugriff auf die entsprechenden Objekte und Aufruf der entsprechenden Methode gesetzt werden.

Diese sind

Eigenschaft	Beschreibung
`invisible`	das entsprechende Objekt ist unsichtbar
`draft`	das entsprechende Objekt wird im Entwurfsmodus dargestellt
`nolabel`	das entsprechende Objekt erhält kein Label

Gesetzt werden diese Eigenschaften direkt beim Anlegen des Objekts, indem das jeweilige Schlüsselwort (bzw. die jeweiligen Schlüsselworte, auch eine Kombination davon ist möglich), durch Leerzeichen getrennt, hinter dem Konstruktionsbefehl noch vor dem zugehörigen Semikolon, geschrieben wird, d.h.

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

3 Zugriff auf Elemente

Der Zugriff auf die Elemente nach dem Konstruieren ist möglich über:

Element	Beschreibung
`constr.points[i]`	liefert den i-ten Punkt oder Gleiter der Konstruktion `constr`, dabei sind auch Mittel- bzw. Teilpunkte unter den Punkten
`constr.lines[i]`	liefert die i-te Gerade, Halbgerade oder Strecke der Konstruktion `constr`, dabei sind auch Parallelen und Senkrechten unter den Geraden
`constr.circles[i]`	liefert den i-ten Kreis der Konstruktion `constr`
`constr.intersections[i]`	liefert den i-ten Schnittpunkt der Konstruktion `constr`
`constr.angles[i]`	liefert den i-ten Winkel der Konstruktion `constr`
`constr.functions[i]`	liefert die i-te Funktion der Konstruktion `constr`
`constr.texts[i]`	liefert das i-te Textelement der Konstruktion `constr`
`constr.polygons[i]`	liefert das i-te Polygon der Konstruktion `constr`
`constr.A`	liefert das Element mit dem Namen `A` der Konstruktion `constr`

4 Makros

Zusätzlich können Makros definiert werden. Schlüsselwort ist `Marco`, die Parameter werden, durch Komma getrennt, in runden Klammern übergeben, der Inhalt innerhalb von geschweiften Klammern. Links vom Zuweisungsoperator kann ein beliebiger Name für das Makro übergeben werden. Die entsprechende Syntax ist also

```
macroName = Macro(param1, param2, param3, ...) { Befehl1; Befehl2; Befehl3; ... };
```

Aufgerufen wird das Makro dann mit

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

5 Beispiel

Ein Beispiel soll zur Veranschaulichung der Anwendung dienen.

```
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');
```