JessieScript Referenz (Version Juni 2010 v0.82)

1 Konstruieren

Einfache mathematische JSXGraph-Konstruktionen können in der von JSXGraph lesbaren Syntax JessieScript 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.

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 Objektnamen 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 Setzen von Eigenschaften

Möchte man Eigenschaften der erzeugten Elemente im Nachhinein verändern, ist das auch per JessieScript möglich. Die entsprechende Syntax lautet

```
objektname.eigenschaft = wert;
```

Ein Beispiel wäre also

```
A(1,2); A.size = 8;
```

Mögliche Eigenschaften sind dabei

Eigenschaft	Beschreibung
`strokecolor`	Linienfarbe, entweder als englischer HTML-Farbname oder als Hex-Angabe #rrggbb
`fillcolor`	Füllfarbe, entweder als englischer HTML-Farbname oder als Hex-Angabe #rrggbb
`highlightstrokecolor`	Linienfarbe während das Objekt hervorgehoben ist, entweder als englischer HTML-Farbname oder als Hex-Angabe #rrggbb
`highlightfillcolor`	Füllfarbe während das Objekt hervorgehoben ist, entweder als englischer HTML-Farbname oder als Hex-Angabe #rrggbb
`labelcolor`	Farbe des Labels, entweder als englischer HTML-Farbname oder als Hex-Angabe #rrggbb
`strokewidth`	Linienstärke, in Pixel
`dash`	Strichelung der Linie, mögliche Werte sind dabei: 0: durchgezogene Linie 1: gepunktete Linie 2: gestrichelte Linie mit kurzen Strichen 3: gestrichelte Linie mit normalen Strichen 4: gestrichelte Linie mit langen Strichen 5: gestrichelte Linie mit abwechselnd normalen und langen Strichen und großen Lücken 6: gestrichelte Linie mit abwechselnd normalen und langen Strichen und kleinen Lücken
`visible`	Objekt wird angezeigt (true) oder versteckt (false)
`shadow`	Objekt bekommt einen Schatteneffekt (true) oder nicht (false)
`size`	(nur für Punkte) Größe des Punktes, in Pixel
`face`	(nur für Punkte) Aussehen des Punktes, mögliche Werte sind dabei: Kreuz: `cross` oder `x` Plus: `plus` oder `+` Kreis: `circle` oder `o` Quadrat: `square` oder `[]` Diamant: `diamond` oder `<>` Dreieck nach oben: `triangleup` oder `A` Dreieck nach unten: `triangledown` oder `v` Dreieck nach rechts: `triangleright` oder `>` Dreieck nach links: `triangleleft` oder `<`

4 Makros

Zusätzlich können Makros definiert werden. Schlüsselwort ist `Macro`, 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,...);
```