jAnimation 2.2

Grundbefehle:

I. **Rechteck(Name des Objekts, [Name des Objekts in das eingefügt werden soll]):** Erstellt ein unformatiertes Rechteck. Dieses ist standardmäßig schwarz und hat die Seitenlängen 100 Pixel zu 100 Pixel. Optional kann man das Rechteck in ein anderes existierendes Objekt einfügen.

II. **Kreis(Name des Objekts, [Name des Objekts in das eingefügt werden soll]):** Erstellt einen unformatierten Kreis. Dieser ist standardmäßig schwarz und hat einen Durchmesser von 100 Pixeln. Optional kann man den Kreis in einen anderen existierenden Kreis einfügen.

III. **Bild(Name des Objekts, [Name des Objekts in das eingefügt werden soll]):** Erstellt ein Bildobjekt. Es stellt nur ein Bild dar. Die Bild-Quelle kann mit „ObjektQuelle()“ geändert werden. Anders als bei Rechtecken und Kreisen ist das Bild bei diesem Objekt keine Textur, es kann also auch verformt werden!

IV. **Video(Name des Objekts, [Name des Objekts in das eingefügt werden soll]):** Erstellt ein Video. Wie bei Bild-Objekten wird auch hier die Quelle, also die Videodatei, mit „ObjektQuelle()“ angegeben. Videos lassen sich den speziellen Video-Befehlen manipulieren. Sie werden später erläutert.

V. **Sequenz(Name des Objekts, [Name des Objekts in das eingefügt werden soll]):** Erstellt eine Sequenz. Sequenzen sind im Prinzip Bilder. Allerdings sind sie Folgen solcher. So kann man mit nur einem Objekt den Eindruck einer Bewegung durch schnelle Bildfolgen schaffen. Wie dies funktioniert wird bei dem Sequenzen erörtert.

VI. **Liste(Zahl bzw. Text oder Objekt, Zahl bzw. Text oder Objekt, ...):** Erstellt eine Liste von Zahlen, Texten oder Objekten. Eine erstellte Liste sollte immer in einen Platzhalter geschrieben werden. Elemente aus der Liste lassen sich mit „platzhalterMitListe[Index]“ auslesen. Anstatt von Index steht eine Zahl, sie gibt an das wievielte Element ausgelesen werden soll. Die Zählung beginnt hierbei allerdings mit 0 und nicht mit 1. Listen lassen sich nachträglich mit den später erwähnten Listen-Befehlen manipulieren.

VII. **Warten(Zeit in Sekunden, [Funktion]):** Führt den angegebenen Code nach der angegebenen Zeit aus. Wenn keine Funktion angegeben wird, pausiert der gesamte folgende Code. Gibt ein Timer-Objekt zurück, das spezifisch gestoppt werden kann.

VIII. **Wiederholen(Zeit in Sekunden, Funktion):** Der selbe Effekt wie bei „Warten()“, allerdings startet der Timer, nachdem er abgelaufen ist, automatisch neu und es entsteht eine Endlos-Schleife. Hier muss eine Funktion definiert werden, da „Wiederholen()“ nicht wie eine Stoppmarke (siehe „Warten()“) zu bewerten ist.

IX. **Stoppen([Timer-Objekt]):** Stoppt alle aktiven Timer. Also alle „Warten()“ und „Wiederholen()“ Befehle. Optional kann ein bestimmter Timer gestoppt werden.

X. **Zufall(Minimum, Maximum):** Generiert eine zufällige ganze Zahl, die zwischen den beiden angegebenen Zahlen liegt. Der Befehl kann als Platzhalter-Wert, aber auch wie ein Parameter in einer Funktion verwendet werden.

XI. **Runden(Zahl):** Rundet die Zahl kaufmännisch auf eine ganze Zahl. Der Befehl kann als Platzhalter-Wert, aber auch wie ein Parameter in einer Funktion verwendet werden.

XII. **Meldung(Text):** Gibt eine Meldung mit dem angegebenen Text.

XIII. **Eingabe(Text, Standard-Wert, Funktion bei Bestätigung, [Sonst Funktion]):** Öffnet einen Dialog, in dem eine Texteingabe möglich ist. Wenn der Dialog bestätigt wird, wird die erste Funktion aufgerufen. Wenn nicht, wird die zweite ausgeführt. Innerhalb der

Funktionen steht die Variable „Wert“ zur Verfügung. Sie enthält die Texteingabe des Benutzers. Der Standard-Wert steht als voreingegebener Text im Textfeld.

XIV. **Abfrage(Text, Funktion bei Bestätigung, [Sonst Funktion]):** Zeigt ein Abfrage-Fenster, das mit Ja oder Nein geschlossen werden kann. Der zweite Parameter gibt die Reaktion bei Bestätigung des Dialogs an. Optional kann noch eine Funktion für den Abbruch definiert werden.

XV. **Schrift(Name des Objekts in das eingefügt werden soll, Text):** Fügt den angegebenen Text in ein Rechteck- bzw. Kreis-Objekt ein. Wenn in dem Objekt schon Text steht, wird dieser ersetzt.

XVI. **Verweis(Dateiname eines Programms):** Öffnet das angegebene Programm aus dem Ordner „meineDateien“. Es muss Dateiname und Dateiendung angegeben werden.

XVII. **Einbinden(Dateiname eines Programms):** Führt die Befehle in der Datei aus, als ob diese in der aufrufenden Datei stehen. Es kann auch auf Objekte und Platzhalter aus der eingebundenen Datei zugegriffen werden.

XVIII. **Internet(URL, [Neues Fenster]):** Öffnet eine Webseite im Internet. Setzt man „Neues Fenster“ auf den Wert „neu“, wird die Seite in einem separaten Browserfenster geöffnet. Beim Standardwert „normal“ ist dies nicht der Fall.

XIX. **Neuladen():** Startet das aktuelle Programm erneut.

XX. **Funktion(Funktion):** Führt die angegebene Funktion aus. Besonders praktisch, wenn die Funktion in einem Platzhalter liegt. So kann ein mehrfach vorkommender Programmbereich beliebig oft an beliebiger Position ausgeführt werden.

XXI. **Schleife(Durchläufe, Funktion):** Führt die angegebene Funktion mehrfach ohne Wartezeiten aus. Es muss angegeben werden wie oft die Funktion ausgeführt werden soll, da eine Endlosschleife bei einer so hohen Wiederholungsgeschwindigkeit den Arbeitsspeicher überfüllen würde. Innerhalb der Funktion kann der aktuelle Durchlauf abgefragt werden, indem man den Platzhalter „Durchlauf“ abruft. Dieser darf dann natürlich nicht geändert werden, da sonst die Anzahl von Durchläufen nicht mehr korrekt ausgeführt werden kann.

Befehle für Objekte:

I. **ObjektName(Objekt bzw. Gruppenname, Name):** Ändert den Namen des angegebenen Objekts auf den neuen. Es kann hier kein Gruppenname und keine Menge von Objekten angegeben werden, denn Namen sind immer einzigartig!

II. **ObjektGruppe(Objekt(e) bzw. Gruppenname(n), Gruppenname):** Setzt für das oder die angegebenen Objekte eine Gruppe. Sollte eine Gruppe angegeben werden, werden alle enthaltenen Objekte der neuen Gruppe hinzugefügt und die alte damit gelöscht.

III. **ObjektPosition(Objekt bzw. Gruppenname, x-Achse, y-Achse):** Positioniert ein Objekt auf der Seite. Nullpunkt des Koordinatensystems ist in der oberen linken Ecke.

IV. **ObjektPositionUm(Objekt bzw. Gruppenname, x-Achse, y-Achse):** Ändert die Koordinaten um die angegebenen Werte. Wenn ein Objekt z.B. Auf (1|2) liegt und die angegebenen Werte (3|4), liegt das Objekt nachher auf (4|6).

V. **ObjektGroesse(Objekt bzw. Gruppenname, Breite, Höhe):** Ändert die Größe eines Objekts.

VI. **ObjektSkalierung(Objekt bzw. Gruppenname, Faktor):** Multipliziert die Seitenlängen des oder der angegebenen Objekte mit dem Faktor. Dies sorgt für eine proportionale Streckung. Schrift-Formatierungen und Liniendicke wird auch vergrößert. Der Streckpunkt liegt in der oberen linken Ecke.

VII. **ObjektQuelle(Objekt bzw. Gruppenname, Farbe oder Bild- oder Videodatei, [Hintergrundbild nur bei Rechteck und Kreis Objekten]):** Gibt einem Kreis- oder Rechteck-Objekt eine Farbe und optional auch eine Textur. Bild-Objekte erhalten eine direkte Bild-Quellenangabe, Video-Objekte eine Video-Quelle. Sequenzen erhalten ebenfalls ein Bildobjekt. Dieses enthält allerdings viele verschiedene Bilder, die alle ohne Abstand exakt nebeneinander stehen müssen. Ein Beispiel für ein Sequenzbild:

VIII. **ObjektFarbe(Objekt bzw. Gruppenname, Farbe, [Hintergrundbild nur bei Rechteck und Kreis Objekten]):** Macht das selbe wie der „ObjektQuelle()“-Befehl, nur dass er speziell für das Kolorieren von Objekten ist. Dieser Befehl eigentlich überflüssig, allerdings ist „ObjektFarbe()“ für einen Anfänger, der nur ein Rechteck rot anmalen will leichter zu begreifen als „ObjektQuelle()“ dafür zu verwenden.

IX. **ObjektLinie(Objekt bzw. Gruppenname, Farbe der Linie, Dicke der Linie, [Linienstil]):** Zeichnet eine Linie um ein Objekt. Für den optionalen Parameter „Stil“ sind diese Werte möglich: gerade, gestrichelt, gepunktet und 3D.

X. **ObjektNeigung(Objekt bzw. Gruppenname, Winkel):** Rotiert ein Objekt am Mittelpunkt. Der Winkel ist als Zahl in Grad anzugeben. (Kein °-Zeichen!)

XI. **ObjektEcken(Objekt bzw. Gruppenname, Radius):** Rundet die Ecken eines Rechtecks ab. Bei Kreisen funktioniert dieser Befehl nicht!

XII. **ObjektAbstaende(Objekt bzw. Gruppenname, Außenabstand, Innenabstand):** Macht um oder in ein Objekt einen unsichtbaren Abstand.

XIII. **ObjektSichtbar(Objekt bzw. Gruppenname, Sichtbarkeit):** Macht ein Objekt sichtbar bzw. unsichtbar. Der angegebene Wert darf zwischen 0 und 1 liegen. Ungerade Zahlen sorgen für teilweise transparente Objekte. Dieser Befehl wirkt sich auch auf Schrift und Objekte in dem Objekt aus!

XIV. **ObjektMaus(Objekt bzw. Gruppenname, Mauszeiger):** Ändert den Mauszeiger des angegebenen Objekts. Mögliche Werte sind: „normal, klick, fadenkreuz, bewegen und warten“.

XV. **ObjektBewegen(Objekt bzw. Gruppenname, Dauer der Bewegung in Sekunden, x-Achse Ziel, y-Achse Ziel, [Funktion nach Ende der Animation], [Geschwindigkeitskurve]):** Bewegt ein Objekt an eine andere Position.

XVI. **ObjektBewegenUm(Objekt bzw. Gruppenname, Dauer der Bewegung in Sekunden, x-Achse Ziel, y-Achse Ziel, [Funktion nach Ende der Animation], [Geschwindigkeit]):** So wie „ObjektBewegen()“, allerdings wird das Objekt von der aktuellen Position um die angegebenen Werte bewegt.

XVII. **ObjektVerformen(Objekt bzw. Gruppenname, Dauer der Verformung in Sekunden, Neue Breite, Neue Höhe, [Funktion nach Ende der Animation], [Geschwindigkeit]):** Ändert die Größe eines Objekts. Bei Kreisen wird die Angabe der Höhe ignoriert, daher kann der Wert dort freigelassen werden.

XVIII. **ObjektSkalieren(Objekt bzw. Gruppenname, Dauer der Animation in Sekunden, Faktor, [Funktion nach Ende der Animation], [Geschwindigkeit]):** Hat den selben Effekt wie „ObjektSkalierung()“, tut dies allerdings flüssig als Animation.

XIX. **ObjektVerblassen(Objekt bzw. Gruppenname, Dauer des Verblassens in Sekunden, Neue Sichtbarkeit, [Funktion nach Ende der Animation], [Geschwindigkeit]):** Lässt ein Objekt langsam sichtbar bzw. unsichtbar werden. Die Sichtbarkeit wird so wie bei dem Befehl „ObjektSichtbar()“ definiert.

XX. **ObjektStoppen(Objekt bzw. Gruppenname):** Stoppt alle aktiven Animationen auf dem aktuellen Stand. Danach können natürlich wieder neue definiert werden.

XXI. **ObjektKlick(Objekt bzw. Gruppenname, Funktion bei Aktionseintritt):** Führt die angegebene Funktion bei einem Mausklick (d.h. Drücken & Loslassen der rechten Maustaste) auf das gesetzte Objekt aus.

XXII. **ObjektKlickStart(Objekt bzw. Gruppenname, Funktion bei Aktionseintritt):** Führt die angegebene Funktion aus, wenn ein Mausklick auf das gesetzte Objekt begonnen wird (d.h. Drücken der rechten Maustaste).

XXIII. **ObjektDoppelKlick(Objekt bzw. Gruppenname, Funktion bei Aktionseintritt):** Führt die angegebene Funktion bei einem doppelten Mausklick auf das gesetzte Objekt aus.

XXIV. **ObjektBeruehren(Objekt bzw. Gruppenname, Funktion bei Aktionseintritt):** Führt die angegebene Funktion aus, wenn die Maus das Objekt berührt.

XXV. **ObjektVerlassen(Objekt bzw. Gruppenname, Funktion bei Aktionseintritt):** Führt die angegebene Funktion aus, wenn die Maus das Objekt nach einer Berührung wieder verlässt.

XXVI. **ObjektStreichen(Objekt bzw. Gruppenname, Funktion bei Aktionseintritt):** Führt die angegebene Funktion oft hintereinander aus, solange die Maus auf dem Objekt ist. Dies kann zu sehr vielen Funktionsaufrufen führen!

XXVII. **ObjektDublizieren(Objekt bzw. Gruppenname, Namensanhang):** Verdoppelt alle angegebenen Objekte. Position, Größe, Hintergrund und sonstige Eigenschaften werden übernommen. Die Namen und Gruppen der Objekte bleiben gleich, es wird allerdings jeweils der angegebene Anhang dahinter geschrieben.
(„testObjekt“ wird bei dem Anhang „-Kopie“ zu „testObjekt-Kopie“)

XXVIII. **ObjektVorne(Objekt bzw. Gruppenname):** Verschiebt das Objekt auf der „z-Achse“ ganz nach vorne. Alle anderen Objekte liegen dann dahinter.

XXIX. **ObjektHinten(Objekt bzw. Gruppenname):** Arbeitet so wie „ObjektVorne()“, verschiebt das Objekt allerdings ganz nach hinten.

XXX. **ObjektEntfernen(Objekt bzw. Gruppenname):** Entfernt das angegebene Objekt.

Befehle für Schrift und Text:

I. **SchriftArt(Objekt bzw. Gruppenname, Schriftart):** Ändert die Schriftart der Texte im angegebenen Objekt.

II. **SchriftGroesse(Objekt bzw. Gruppenname, Größe):** Ändert die Schriftgröße der Texte im angegebenen Objekt.

III. **SchriftFarbe(Objekt bzw. Gruppenname, Farbe):** Ändert die Schriftfarbe der Texte im angegebenen Objekt.

IV. **SchriftAusrichtung(Objekt bzw. Gruppenname, Ausrichtung):** Ändert die Textausrichtung der Texte im angegebenen Objekt. Es gibt: links, zentriert, rechts und block.

V. **SchriftDicke(Objekt bzw. Gruppenname, Dicke):** Ändert die Schrift Dicke. Möglich Werte sind „fett“ und „normal“.

VI. **SchriftZeilenhoehe(Objekt bzw. Gruppenname, Zeilenhöhe):** Ändert die Höhe der Zeilen aller Texte im angegebenen Objekt.
VII. **SchriftHintergrund(Objekt bzw. Gruppenname, Farbe, [Hintergrundbild]):** Anzuwenden wie „ObjektHintergrund“. Ändert die Texthinterlegung aller Texte im angegebenen Objekt.
VIII. **SchriftSichtbar(Objekt bzw. Gruppenname, Sichtbarkeit):** Anzuwenden wie „ObjektSichtbar()“. Ändert die Sichtbarkeit aller Texte im angegebenen Objekt.
IX. **SchriftEntfernen(Objekt bzw. Gruppenname):** Entfernt alle Texte im angegebenen Objekt.

Befehle für den Zwischenspeicher:

I. **SpeicherSchreiben(Speichername, Wert):** Speichert den angegebenen Wert für 30 Tage im Zwischenspeicher zwischen.Wenn der Wert während dieser Frist neu gesetzt wird, werden die 30 Tage vom aktuellen Tag aus neu gesetzt. Die Funktion erfordert aktivierte Cookies im Browser.
II. **SpeicherLesen(Speichername):** Gibt den Wert, der für den angegebenen Namen abgelegt wurde zurück. Genau wie „SpeicherSchreiben()“ erfordert auch dieser Befehl aktivierte Cookies.

Befehle für Soundeffekte:

I. **Musik(Dateiname, [Wiederholung], [Funktion bei Abschluss]):** Spielt die angegebene Audiodatei ab. Setzt man die Wiederholung auf „endlos“, spielt die Musik endlos. Bei dem Wert „normal“ nur einmal, dies ist Standard. Die Funktion wird ausgeführt, sobald die Musik zu Ende gespielt hat. Der Befehl gibt außerdem ein Musikobjekt für weitere Verwendungszwecke zurück, siehe weitere Befehle.
II. **MusikStoppen(Musikobjekt):** Stoppt das angegebene Musikobjekt.
III. **MusikStumm(Musikobjekt):** Das angegebene Musikobjekt spielt lautlos weiter.
IV. **MusikLaut(Musikobjekt):** Ein stummes Musikobjekt spielt wieder laut.
V. **MusikDeaktivieren():** Die Musikfunktion wird komplett eingefroren. Ein lässt sich keine Musik mehr starten oder auch stoppen.
VI. **MusikAktivieren():** Die Musikfunktion wird wieder aktiviert.

Befehle für Video-Objekte:

I. **VideoStoppen(Objekt bzw. Gruppenname):** Stoppt ein spielendes Video.
II. **VideoAbspielen(Objekt bzw. Gruppenname):** Startet ein gestopptes Video erneut und setzt an der letzten Abspielposition wieder ein.
III. **VideoBeendet(Objekt bzw. Gruppenname, Funktion):** Funktion wird ausgeführt, sobald das angegebene Video-Objekt zu Ende gespielt hat.

Befehle für Sequenz-Objekte:

I. **SequenzPosition(Objekt bzw. Gruppenname, Position):** Zeigt im Sequenzobjekt das Bild aus der Bildfolge an der angegebenen Position.
II. **SequenzLaenge(Objekt bzw. Gruppenname, Länge):** Setzt die Zahl der Einzelbilder die in dem „Bildfolge“-Bild zu finden sind.
III. **SequenzGeschwindigkeit(Objekt bzw. Gruppenname, Geschwindigkeit):** Setzt sie Zeit in Sekunden, die den Intervall zwischen den einzelnen Bildern der Folge markieren.

IV. **SequenzWiederholen(Objekt bzw. Gruppenname, Wiederholen):** Setzt, ob das Sequenzobjekt die Bildfolge endlos wiederholen soll (Wert = 1), oder nur einmal durchlaufen soll (Wert = 0).
V. **SequenzStarten(Objekt bzw. Gruppenname, [Funktion nach Ende]):** Startet eine Sequenz, nachdem sie initialisiert wurde. Die Funktion wird aufgeführt, nachdem die Sequenz durchgelaufen ist. In endlosen Sequenzen passiert das nie.
VI. **SequenzStoppen(Objekt bzw. Gruppenname):** Stoppt eine gestartete Sequenz.

Befehle für Listen:

I. **ListeLaenge(Liste):** Gibt die Anzahl der Elemente in der angegebenen Liste zurück.
II. **ListeErweitern(Liste, Zahl bzw. Text oder Objekt):** Erzeugt eine neue Liste, die gleich der angegebenen ist. An die neue Liste wird der Inhalt des zweiten Parameters als neues Element angehängt. Die neue Liste wird zurückgegeben.
III. **ListeVerbinden(Liste, weitere Liste):** Gibt eine neue Liste zurück. Diese enthält alle Elemente der ersten und zweiten Liste. Die Listen werden also aneinander gehängt. Die Indizes werden komplett neu vergeben. Beachten Sie, dass die zweite Liste hinten an die erste angehängt wird.
IV. **ListeLoeschen(Liste, Index):** Entfernt das Objekt am angegebenen Index aus der Liste und gibt die neu entstandene Liste zurück. Alle Indizes werden dabei neu vergeben!
V. **ListeAusText(Text, Trenner):** Gibt eine Liste zurück. Die Listenelemente entstehen, indem der angegebene Text vom Trenner-Text „gespalten wird“.
VI. **ListeZuText(Liste, Bindeglied):** Gibt einen Text zurück. Dieser Text enthält alle Listenelemente. Sie sind alle aneinander gehängt. Das Bindeglied steht dazwischen.

Befehle zum Caching von Dateien:

I. **VorladenBilder(Listen-Element):** Lädt alle angegebenen Bilder der Liste in den Cache, so dass diese ohne Ladezeit dargestellt werden können. Nach diesem Befehl folgender Code wird erst ausgeführt, wenn alle Bilder geladen sind.
II. **VorladenMusik(Listen-Element):** Arbeitet so wie der „VorladenBilder()“-Befehl, speichert allerdings Musik-Dateien zwischen.
III. **VorladenVideos(Listen-Element):** Auch dieser Befehl lädt vor, diesmal allerdings Video-Dateien.

Befehle für die Tastatur:

I. **TastaturDruck(Taste, Funktion):** Führt die angegebene Funktion aus, wenn die angegebene Taste gedrückt wird. Bitte geben Sie nur eine der sieben unten aufgelisteten Tasten an.
II. **TastaturHalten(Taste, Funktion):** Führt die angegebene Funktion aus, solange die angegebene Taste gedrückt wird. Bitte geben Sie nur eine der sieben unten aufgelisteten Tasten an.
III. **TastaturLoslassen(Taste, Funktion):** Führt die angegebene Funktion aus, wenn die angegebene Taste, nach dem Drücken, wieder losgelassen wird. Bitte geben Sie nur eine der sieben unten aufgelisteten Tasten an.

Folgende Tasten können angegeben werden: **„Escape“** (Esc-Taste), **„Enter“** (Enter-Taste), **„Leertaste“** (Leertaste), **„Links“** (linke Pfeiltaste), **„Rechts“** (rechte Pfeiltaste), **„Oben“** (obere Pfeiltaste), **„Unten“** (untere Pfeiltaste). Außerdem die Zeichen von **„0-9“** und **„A-Z“**. Um jede beliebige Tastenaktion zu erfassen gibt man den Wert **„Immer“** an.

Befehle für logische Überprüfungen & Konstrukte:

I. **WennGleich(Wert oder Platzhalter 1, Wert oder Platzhalter 2, Funktion bei Übereinstimmung, [Funktion bei Ungleichheit]):** Führt die angegebene Funktion aus, wenn beide Werte identisch sind. Alle Wertetypen können verglichen werden! Optional kann eine Funktion angegeben werden, die in allen anderen Fällen als der Gleichheit aufgerufen wird.

II. **WennGroesser(Wert oder Platzhalter 1, Wert oder Platzhalter 2, Funktion bei größerem Wert, [Funktion für sonstige Fälle]):** Führt die angegebene Funktion aus, wenn der erste Wert größer als der zweite ist. Sonst identisch mit dem „WennGleich()“-Befehl.

III. **WennKleiner(Wert oder Platzhalter 1, Wert oder Platzhalter 2, Funktion bei kleinerem Wert, [Funktion für sonstige Fälle]):** Führt die angegebene Funktion aus, wenn der erste Wert kleiner als der zweite ist. Sonst identisch mit dem „WennGleich()“-Befehl.

IV. **WennNichtGleich(Wert oder Platzhalter 1, Wert oder Platzhalter 2, Funktion bei ungleichem Wert, [Funktion für sonstige Fälle]):** Führt die angegebene Funktion aus, wenn der erste Wert nicht identisch mit dem zweiten ist. Sonst so anzuwenden wie der „WennGleich()“-Befehl.

V. **WennBeruehrt(Objekt 1, Objekt 2, Funktion bei Überlagerung, [Funktion für sonstige Fälle]):** Führt die angegebene Funktion aus, wenn die die Objekte ganz oder teilweise übereinander auf der „Wand“ (siehe Erläuterungen) liegen.

VI. **Wenn(Ausdruck, Dann-Funktion, [Funktion für sonstige Fälle]):** Wenn vereinigt alle anderen separaten Befehle. In den Ausdruck kann ein beliebig komplexes Konstrukt aus Überprüfungen aufgebaut werden. Zum besseren Verständnis sind hier nun einige Beispiele zur Erklärung, da diese besser zu verstehen sind als Beschreibungen:

- `Wenn(1+1 gleich 2, { Meldung(1) })`
- `Wenn(„rechteck“ beruehrt „kreis“, { Meldung(1) })`
- `Wenn(10 groesser 5, { Meldung(1) })`
- `Wenn(platzhalter kleiner MathePi, { Meldung(1) })`
- `Wenn(MatheWurzel(4) ungleich 3, { Meldung(1) })`
- `Wenn(4*4 gleich 16 und 9*3 ungleich 27, { Meldung(1) })`
- `Wenn(„rechteck“ beruehrt „kreis“ oder platzhalter gleich 1, { Meldung(1) })`
- `Wenn(a+b gleich c und c-b groesser a und c-a ungleich c, { Meldung(1) })`

Diese Operatoren gibt es: **„und“** (UND-Verknüpfung), **„oder“** (ODER-Verknüpfung), **„gleich“** (Gleichheit), **„ungleich“** (keine Gleichheit), **„groesser“** (Groesser-Vergleich), **„kleiner“** (Kleiner-Vergleich), **„beruehrt“** (Siehe „WennBeruehrt()“).

Mathematische Funktionen & Konstanten:

I. **MatheWurzel(Radikand, [Wurzelexponent]):** Die nte-Wurzel der Zahl. Wenn n (Wurzelexponent) nicht gegeben ist, ist dieser 2.

II. **MathePotenz(Basis, Exponent):** Gibt die Potenz beider Werte.

III. **MatheLogarithmus(Numerus, Basis):** Gibt den Logarithmus aus beiden Zahlen zurück.

IV. **MatheSinus(Winkel):** Gibt den Sinus des Winkels (Grad).

V. **MatheCosinus(Winkel):** Gibt den Cosinus des Winkels (Grad).

VI. **MatheTangens(Winkel):** Gibt den Tangens des Winkels (Grad).
VII. **MatheArcusSinus(Winkel):** Gibt den Arcussinus des Winkels (Grad).
VIII. **MatheArcusCosinus(Winkel):** Gibt den Arcuscosinus des Winkels (Grad).
IX. **MatheArcusTangens(Winkel):** Gibt den Arcustangens des Winkels (Grad).
X. **MathePi:** Platzhalter enthält den Wert von Pi.
XI. **MatheE:** Platzhalter enthält die eulersche Konstante.

Platzhalter:

Ein Platzhalter hat immer einen Namen, der einen Wert im Programm repräsentiert. Dieser Wert kann eine Zahl, aber auch Text sein. Der Wert kann sich während der Laufzeit des Programms ändern. Einen Platzhalter definiert man z.B. so: `meinPlatzhalter = 1`. Es wird ein Platzhalter mit dem Namen „meinPlatzhalter“ definiert. Wichtig ist, dass ein Platzhalter immer ein Wort ohne Sonderzeichen (wie z.B. Umlaute) sein muss. Der Wert des Platzhalters ist in diesem Fall 1. Nun kann man in alle Befehle anstatt einer Zahl, oder einem Text, den Platzhalter einfügen. Der Platzhalter darf dabei nie in Anführungszeichen angegeben werden, da er sonst nur ein normaler Text wäre. An die Stelle eines Platzhalters tritt dann immer der dahinter liegende Wert. Einen Platzhalter-Wert ändert man, indem man einfach noch einmal einen Platzhalter mit dem selben Namen definiert, der alte Platzhalter wird dann ersetzt.
Noch ein Tipp: Um einfache Rechenoperationen wie +1 oder -1 anzuwenden kann man auch `meinPlatzhalter++` bzw. `meinPlatzhalter--` schreiben.

Belegte Platzhalter:

Manche Platzhalter dürfen nicht verwendet werden bzw. Haben bereits praktische Werte:

I. **Befehlsnamen:** Platzhalter dürfen nicht die Namen von Befehlen tragen.
II. **Systemkonstanten:** Dieser Platzhalter ist belegt: „kernel“. Bei Änderungen kann das Programm unbrauchbar werden. Wobei auch der Platzhalter „Durchlauf“ bei Änderung im Zusammenhang mit Schleifen Probleme verursachen wird.
III. **Aufrufer:** Repräsentiert in allen Funktionen von Animationen und auch Benutzeraktionen das Element, das die Funktion aufgerufen hat. Dies ist immer dann nützlich, wenn der Objektname aus Platzhaltern besteht und damit flexibel ist. Denn so kann eindeutig ein Element identifiziert werden, auch wenn es durch die Platzhalterzusammensetzung schon nicht mehr selektiert werden kann, da die Platzhalterwerte sich geändert haben.
IV. **MausLinks:** Dieser Platzhalter kann nur in Funktionen, die von Objekten aufgerufen wurden verwendet werden (Animationen, Mausaktionen). Beinhaltet die x-Koordinate der Maus. Dieser Platzhalter kann nicht geändert werden!
V. **MausOben:** So wie „MausLinks“. Beinhaltet aber die y-Koordinate der Maus.
VI. **Mathematische Konstanten:** Die Platzhalter „MathePi“ und „MatheE“ sind belegt. Mehr darüber bei den mathematischen Funktionen & Konstanten.

Erläuterungen:

I. **Regeln:** In jAnimation können alle Operationen und Befehle aus JavaScript genutzt werden. Um komplexe Aufgaben zu lösen, sollte man diese beherrschen.
II. **Funktionen:** Funktionen in jAnimation-Befehlen verwendet man folgendermaßen: `{ ...Code... }`. An die Position von „...Code...“ kann man beliebig viele andere Befehle, mit Semikolon bzw. Leerzeilen getrennt, einfügen.
III. **„Geschwindigkeit“ bei Animationen:** Die Geschwindigkeitskurve bei Animationen ist standardmäßig linear, also gleichmäßig. Diese lässt sich allerdings auch ändern.

Folgende Arten sind möglich: „beschleunigen“, „entschleunigen“, „einwackeln“, „auswackeln“, „aufprallen“ und der Standardwert „linear“. Die Be-/Entschleunigung entspricht den physikalischen Gesetzen und eignet sich somit auch zur Simulation.

IV. **Mehrere Objekte:** In allen Befehlen mit Objekten kann mehr als ein Objekt angegeben werden! Die Objekte schreibt man dann durch ein Komma getrennt hintereinander. Besonders wichtig bei dem Befehl „ObjektGruppe()“, denn nur so können die vielen Objekte für die Gruppenbildung angegeben werden. Bitte beachten Sie, dass ein Leerzeichen nach einem angegebenen Komma mit zum darauf folgenden Objektnamen gezählt wird! Daher darf nach dem Komma kein Leerzeichen stehen, es sei denn dieses gehört zum Objektnamen.

V. **Farben:** Farbangaben erfolgen alle auf englisch (nur die 120 Netscape-Farbnamen!), nach dem Hexadezimalen-Farbschema oder in der Form „rgb(rot, grün, blau)“, wobei jeder Wert zwischen 0 und 255 liegen kann.

VI. **Texte:** Alle Werte in Befehlen, die Zeichen, die nicht Zahlen sind, enthalten sind Texte. Texte müssen immer in Anführungszeichen gesetzt werden.

VII. **Eckige Klammern []:** Wenn in der Dokumentation eckige Klammern stehen, heißt das, dass die sich in der Klammern befindlichen Parameter für die Befehle optional sind.

VIII. **Die Wand:** Die weiße Fläche auf der die Objekte platziert werden ist die Wand. Man kann auch die Wand bearbeiten! Dafür muss anstatt eines Objektnamens in Befehle einfach nur das Wort „Wand“ eingesetzt werden. So kann z.B. die Größe („ObjektGroesse()“) oder auch die Farbe („ObjektFarbe()“) dieser verändert werden.

IX. **Alte Werte:** In vielen Befehlen werden mehr als ein Wert eingesetzt. Manchmal möchte man aber nur einen der Werte ändern. Anstatt den alten Wert nochmal einzusetzen, kann auch einfach der Text „X“ eingesetzt werden. Die Anführungszeichen sind notwendig. Sofern man dies nicht in Befehle wie „Meldung()“ oder „Schrift()“ einsetzt, wird an die Stelle von X der alte Wert eingesetzt. So kann zum Beispiel nur die Höhe eines Objektes geändert werden oder die Breite zu definieren oder gar zu wissen, was z.B. bei zufällig eingesetzten Werten mit „Zufall()“ der Fall ist.

X. **Datei-Formate:** Musik-Elemente müssen immer im ogg- (oga), mp3- oder wav-Format gespeichert sein. Im besten Fall sogar in allen dreien gleichzeitig , denn so werden garantiert alle Browser unterstützt. Bei Video-Elementen gibt es auch Einschränkungen: Es muss hierbei immer das mp4-, mov- oder ogg- (ogv) Format verwendet werden. Für Bilder gibt es keine klaren Einschränkungen es sollten aber dennoch nur jpg-, png- und gif-Bilder verwendet werden.

XI. **Befehl-Rückgaben:** Fast alle Befehle geben den Wert, den sie ändert können auch zurück. So kann man gesetzte Werte auch wieder auslesen. Dies geht zum Beispiel so:

```
ObjektGroesse("test", 100, 200)
groesseVonTest = ObjektGroesse("test")
Meldung("Breite: "+groesseVonTest[0]+" und Hoehe: "+groesseVonTest[1])
```

Die 0 bzw. 1 steht für den gewünschten Wert. 0 steht für den ersten Wert nach dem Objektnamen (z.B. Breite), 1 für den zweiten Wert nach dem Objektnamen (z.B. Höhe) und so weiter.