# Script Definitions

## 1. Ereignis-Parameter

### 1.1 Param - self

**Beschreibung:** self Referenz auf das Ereignis, für dass das Skript ausgeführt wird. Beispiel -- Ausgabe des Ereignisnamens print(self.name)

### 1.2 Param - deferredCall

**Beschreibung:** deferredCall Gibt den Namen der Verzögerung an, oder nil, falls das Ereignis nicht verzögert ausgeführt wird. Siehe Funktion defer() für weitere Informationen.

## 2. Globale Variablen/Funktionen

### 2.1 Keyword - $

**Beschreibung:** $ Referenziert ein Objekt oder ein Ereignis/Modul anhand eines Namens. Im Gegensatz zu layout:getEntityByName, was ein Objekt erst zur Laufzeit anhand eines Namens sucht, ist eine Referenz eine fest verdrahtete Verknüpfung, die zum Zeitpunkt der Skript-Bearbeitung eindeutig festgelegt wird. Eine Referenz bleibt auch nachträglich gültig, auch dann wenn der Name des referenzierten Objekts geändert wird oder mehrere Objekte den gleichen Namen besitzen. Referenzen sind vergleichbar mit Speicheradressen und daher schneller in der Verarbeitung als die Suche nach Objekten. Beispiel -- Referenzierung des Objekts mit dem Namen "ICE-Lok" local vehicle = $("ICE-Lok") vehicle.currentSpeed = 0

### 2.2 Keyword - keyword

**Beschreibung:** keyword Repräsentiert einen Wert vom Typ "Schlagwort". Ein Schlagwort ist jede beliebige Objektvariable vom Typ "Schlagwort". Schlagwort-Variablen erlauben die Klassifizierung von Objekten und somit eine einheitliche Verarbeitung, wie z.B. das Auslösen eines Ereignisses für alle Objekte, die ein bestimmtes Schlagwort besitzen. Beispiel -- Hinzufügen des Schlagworts "Sperrgleis" zu dem Objekt track track.variables["Sperrgleis"] = keyword -- Prüfen, ob das Objekt track ein Schlagwort "Sperrgleis" besitzt if track.variables["Sperrgleis"] == keyword then ... end

### 2.3 Var - layout

**Beschreibung:** layout Globale Variable, um auf Eigenschaften und Inhalte der Anlage zuzugreifen. Beispiel local vehicle = layout:getEntityByName("ICE-Lok") local time = layout.time

### 2.4 Var - network

**Beschreibung:** network Ermöglicht den Zugriff auf Netzwerk-Funktionen mit Hilfe der externen Steuerschnittstelle (siehe Wiki für weitere Informationen).

### 2.5 Func - defer

**Beschreibung:** defer (duration, name) Verzögert die Ausführung des Ereignisses um eine angegebene Dauer. Beim Aufruf der Funktion wird das Ereignis gestoppt und nach Ablauf der Dauer erneut von Beginn an ausgeführt, mit den gleichen Parametern wie beim ersten Aufruf. Verzögerungen können genutzt werden, um zeitgesteuerte Abläufe auch ohne Timer zu implementieren. Parameter duration - Dauer der Verzögerung in Sekunden. name - Beliebiger Name, um die Verzögerung zu kennzeichen. Der Name wird beim erneuten Aufruf im Parameter deferredCall übergeben, wodurch zwischen verschiedenen Verzögerungen innerhalb eines Ereignisses unterschieden werden kann. Beispiel if not deferredCall then -- Erster, nicht verzögerter Aufruf des Ereignisses ... -- Ereignis um 5 Sekunden verzögern defer(5, "Verzögerung A") elseif deferredCall == "Verzögerung A" then -- Wird nach 5 Sekunden aufgerufen ... -- Ereignis erneut verzögern defer(23, "Verzögerung B") elseif deferredCall == "Verzögerung B" then -- Wird nach 23 Sekunden aufgerufen ... end

### 2.6 Func - toTime

**Beschreibung:** toTime (x) Konvertiert einen Wert x zu einer Zeitangabe zwischen 0 und 24 Uhr. Parameter x - Entweder ein string im Format "HH:MM" oder eine Zahl zwischen 0 und 1, wobei 0 für Mitternacht und 0.5 für 12 Uhr Mittag steht. Rückgabewerte Gibt ein Objekt vom Typ Zeit zurück. Hinweis Objekte vom Typ Zeit unterstützen die Rechenoperationen + und - sowie die Vergleichsoperationen < und >, wenn beide Werte vom Typ Zeit sind. Zudem werden Zeit-Objekte automatisch bei Bedarf in einen string umgewandelt, in der Form "HH:MM". Beispiel -- Simulationszeit auf 9:45 Uhr setzen layout.time = toTime("9:45") -- Simulationszeit um eine halbe Stunde erhöhen layout.time = layout.time + toTime("0:30") -- Automatische Konvertierung einer Zeitangabe in einen string print("Die aktuelle Simulationszeit lautet: " .. layout.time) -- Prüfung, ob 12 Uhr Mittag vorüber ist if layout.time > toTime(0.5) then

## 3. Anlage

### 3.1 Prop - activeCameras

**Beschreibung:** activeCameras Repräsentiert die aktive Kamera einer Ansicht, wobei der Index die konkrete Ansicht definiert. Die Hauptansicht besitzt den Index 0. Im Falle von Cockpit-Ansichten entspricht die aktive Kamera dem Fahrzeug. Beispiel -- Hauptansicht auf ein Fahrzeug setzen layout.activeCameras[0] = vehicle

### 3.2 Prop - time

**Beschreibung:** time Repräsentiert die aktuelle Simulationszeit, als Typ "Zeitangabe", siehe toTime().

### 3.3 Method - enumEntities

**Beschreibung:** enumEntities (callback) Ruft für alle Objekte in der Anlage eine benutzerdefinierte Funktion auf, wobei das jeweilige Objekt als Parameter übergeben wird. Beispiel -- Alle Objekte ausblenden layout:enumEntities( function (entity) entity.visible = false end )

### 3.4 Method - enumVehicles

**Beschreibung:** enumVehicles (callback) Ruft für alle Fahrzeuge in der Anlage eine benutzerdefinierte Funktion auf, wobei das jeweilige Fahrzeug als Parameter übergeben wird. Beispiel -- Sofortiger Stopp aller Fahrzeuge layout:enumVehicles( function (vehicle) vehicle.currentSpeed = 0 end )

### 3.5 Method - getEntityByName

**Beschreibung:** getEntityByName (name) Sucht ein Objekt anhand eines Namens. Parameter name - Name des Objekts, das zurückgeliefert werden soll. Besitzen mehrere Objekte in der Anlage den gleichen Namen, wird das erste gefundene Objekt zurückgeliefert. Rückgabewerte Gibt das Objekt mit dem angegebenen Namen zurück oder nil, wenn kein Objekt gefunden werden konnte. Hinweis Für konstante Referenzierungen von Objekten wird das Schlüsselwort $ empfohlen.

### 3.6 Method - getEntitiesByName

**Beschreibung:** getEntitiesByName (name) Sucht alle Objekte anhand eines Namens. Parameter name - Name der Objekte, die zurückgeliefert werden sollen. Rückgabewerte Gibt eine Liste aller Objekte mit dem angegebenen Namen zurück.

### 3.7 Method - getEntitiesByKeyword

**Beschreibung:** getEntitiesByKeyword (name) Ermittelt alle Objekte anhand eines Schlagworts. Parameter name - Name des Schlagworts, nach dem gesucht werden soll. Rückgabewerte Gibt eine Liste aller Objekte mit dem angegebenen Schlagwort zurück.

### 3.8 Method - getEntitiesLinkedTo

**Beschreibung:** getEntitiesLinkedTo (target) Gibt eine Liste aller Objekte zurück, die mit einem bestimmten Ziel verknüpft sind. Parameter target - Zielobjekt Rückgabewerte Gibt eine Liste aller verknüpften Objekte zurück.

### 3.9 Method - getRoutesByName

**Beschreibung:** getRoutesByName (name) Ermittelt alle Fahrstraßen anhand eines Namens. Parameter name - Name, nach dem gesucht werden soll. Rückgabewerte Gibt eine Liste aller Fahrstraßen mit dem angegebenen Namen zurück.

## 3.10 Method - getRoutesByKeyword

**Beschreibung:** getRoutesByKeyword (name) Ermittelt alle Fahrstraßen anhand eines Schlagworts. Parameter name - Name des Schlagworts, nach dem gesucht werden soll. Rückgabewerte Gibt eine Liste aller Fahrstraßen mit dem angegebenen Schlagwort zurück.

## 3.11 Method - getVehicleGroup

**Beschreibung:** getVehicleGroup (vehicle, [filter]) Ermittelt alle Fahrzeuge, die zu einem Fahrzeugverbund gehören. Parameter vehicle - Fahrzeug, zu dem der Fahrzeugverbund ermittelt werden soll. filter - Optionale Zahl, um nur bestimmte Fahrzeuge zu ermitteln: 0 - Alle Fahrzeuge 1 - Fahrzeuge mit Antrieb 2 - Fahrzeuge mit aktivem Antrieb 3 - Fahrzeuge ohne Antrieb Rückgabewerte Wert 1: Eine Liste mit allen Fahrzeugen des Fahrzeugverbunds, beginnend mit dem vordersten Fahrzeug (ausgehend von der Fahrtrichtung des Fahrzeugverbunds). Wert 2: Eine Liste mit der Ausrichtung für jedes Fahrzeug, relativ zur Fahrtrichtung des Fahrzeugsverbunds (1 = gleiche Richtung, -1 = entgegengesetzte Richtung).

## 3.12 Method - getVehicleGroupLength

**Beschreibung:** getVehicleGroupLength (vehicle) Ermittelt die Gesamtlänge eines Fahrzeugverbunds. Parameter vehicle - Fahrzeug, dessen Verbundslänge ermittelt werden soll. Rückgabewerte Gibt die Fahrzeugverbundslänge in m zurück.

## 3.13 Method - getVehiclesOn

**Beschreibung:** getVehiclesOn (object) Ermittelt alle Fahrzeuge, die auf einem bestimmten Gleis oder Gleiskontakt liegen. Parameter object - Gleis/Gleiskontakt, auf dem die Fahrzeuge ermittelt werden sollen. Rückgabewerte Gibt eine Liste mit allen Fahrzeugen auf dem Gleis/Gleiskontakt zurück.

## 3.14 Method - getVehiclesOnRoute

**Beschreibung:** getVehiclesOnRoute (contact0, contact1) Ermittelt alle Fahrzeuge, die sich auf der kürzesten Strecke zwischen zwei Gleiskontakten befinden. Parameter contact0 - Gleiskontakt, der den Anfang der Strecke markiert. contact1 - Gleiskontakt, der das Ende der Strecke markiert. Rückgabewerte Gibt eine Liste mit allen Fahrzeugen auf der angegebenen Strecke zurück. Hinweis Die Länge der Strecke zwischen beiden Gleiskontakten darf maximal 150 m betragen. Die Ausrichtung bzw. der Aktivierungszustand der Gleiskontakte hat keinen Einfluss auf die Strecke, es wird der kürzeste Weg zwischen beiden Kontakten geprüft.

## 3.15 Method - getEventsByName

**Beschreibung:** getEventsByName (name) Ermittelt alle Ereignisse/Module anhand eines Namens. Parameter name - Name, nach dem gesucht werden soll. Rückgabewerte Gibt eine Liste aller Ereignisse/Module mit dem angegebenen Namen zurück.

# 4. Objekte

## 4.1 Prop - name

**Beschreibung:** name Repräsentiert den Namen eines Objekts.

## 4.2 Prop - visible

**Beschreibung:** visible Steuert die Sichtbarkeit eines Objekts (Boolean).

## 4.3 Prop - link

**Beschreibung:** link Enthält eine Referenz zu einem Ziel-Objekt, mit dem das Objekt verknüpft ist, oder nil, wenn keine Verknüpfung existiert.

## 4.4 Prop - trackContact

**Beschreibung:** trackContact Ermöglicht den Zugriff auf die Gleiskontaktfunktionen eines Objekts. Ist nil, wenn das Objekt kein Gleiskontakt ist.

## 4.5 Prop - crane

**Beschreibung:** crane Ermöglicht den Zugriff auf die Kranfunktionen eines Objekts. Ist nil, wenn das Objekt kein Kran ist.

## 4.6 Prop - size

**Beschreibung:** size Enthält die Größe eines Objekts als Tabelle mit den Feldern x, y und z (3D-Vektor).

## 4.7 Prop - transformation

**Beschreibung:** transformation Ermöglicht den Zugriff auf die Transformation (Position, Rotation, Skalierung) eines Objekts. Alle Einheiten werden in Meter und Maßstab 1:1 interpretiert.

## 4.8 Prop - variables

**Beschreibung:** variables Ermöglicht den Zugriff auf die Variablen eines Objekts. Beispiel -- Variable hinzufügen/bearbeiten track.variables["Zähler"] = 1 -- Variable löschen track.variables["Zähler"] = nil

## 4.9 Prop - actions

**Beschreibung:** actions Ermöglicht den Zugriff auf die Schalter-Aktionen eines Objekts. Der Index entspricht dem Schalternamen. Wird ein leerer Index übergeben, werden alle Schalter gesteuert. Beispiel -- Einen Schalter mit dem Namen "Licht" aktivieren object.actions["Licht"].state = 1 -- Alle Schalter

deaktivieren object.actions[""].state = 0

### 4.10 Prop - animations

**Beschreibung:** animations Ermöglicht den Zugriff auf die Animationsfunktionen eines Objekts. Der Index entspricht dem Animationsnamen. Wird ein leerer Index übergeben, werden alle Animationen gesteuert. Beispiel -- Eine Animation mit dem Namen "Pantograph" abspielen vehicle.animations["Pantograph"]:play() -- Alle Animationen stoppen vehicle.animations[""]:stop()

### 4.11 Prop - sounds

**Beschreibung:** sounds Ermöglicht den Zugriff auf die Geräusche einer Geräuschquelle. Der Index entspricht dem Geräuschnamen. Wird ein leerer Index übergeben, werden alle Geräusche gesteuert. Beispiel -- Ein Geräusch mit dem Namen "Hupe" abspielen vehicle.sounds["Hupe"]:play() -- Alle Geräusche stoppen vehicle.sounds[""]:stop()

### 4.12 Prop - labels

**Beschreibung:** labels Ermöglicht den Zugriff auf die Beschriftungen eines Objekts. Der Index entspricht dem Beschriftungsnamen. Beispiel -- Eine Beschriftung mit dem Namen "Zugziel" setzen object.labels["Zugziel"].text = "Berlin"

### 4.13 Prop - mute

**Beschreibung:** mute Aktiviert/Deaktiviert die Stummschaltung der Objektgeräusche.

### 4.14 Prop - volume

**Beschreibung:** volume Gibt die Lautstärke der Geräusche zwischen 0 (stumm) und 100 (maximale Lautstärke) an.

## 5. Objekte (Transformation)

### 5.1 Prop - position

**Beschreibung:** position Enthält die Position eines Objekts als Tabelle mit den Feldern x, y und z (3D-Vektor).

### 5.2 Prop - rotation

**Beschreibung:** rotation Enthält die Rotation eines Objekts als Tabelle mit den Feldern x, y, z und w (Quaternion).

### 5.3 Prop - scaling

**Beschreibung:** scaling Enthält die Skalierung eines Objekts als Gleitkommazahl.

## 5.4 Method - reset

**Beschreibung:** reset () Setzt die Position, Rotation und Skalierung auf die Standard-Werte zurück.

## 5.5 Method - resetPosition

**Beschreibung:** resetPosition () Setzt die Position auf den Standard-Wert zurück.

## 5.6 Method - resetRotation

**Beschreibung:** resetRotation () Setzt die Rotation auf den Standard-Wert zurück.

## 5.7 Method - resetScaling

**Beschreibung:** resetScaling () Setzt die Skalierung auf den Standard-Wert zurück.

## 5.8 Method - translate

**Beschreibung:** translate (x, y, z) Veschiebt das Objekt entlang der Koordinatenachsen x, y und z.

## 5.9 Method - translateX

**Beschreibung:** translateX (x) Veschiebt das Objekt entlang der Koordinatenachse x.

## 5.10 Method - translateY

**Beschreibung:** translateY (y) Veschiebt das Objekt entlang der Koordinatenachse y.

## 5.11 Method - translateZ

**Beschreibung:** translateZ (z) Veschiebt das Objekt entlang der Koordinatenachse z.

## 5.12 Method - rotate

**Beschreibung:** rotate (x, y, z, w) Rotiert das Objekt um die Quaternion (x, y, z, w).

## 5.13 Method - rotateX

**Beschreibung:** rotateX (r) Rotiert das Objekt um die Koordinatenachse x mit dem Winkel r (Radiant).

## 5.14 Method - rotateY

**Beschreibung:** rotateY (r) Rotiert das Objekt um die Koordinatenachse y mit dem Winkel r (Radiant).

### 5.15 Method - rotateZ

**Beschreibung:** rotateZ (r) Rotiert das Objekt um die Koordinatenachse z mit dem Winkel r (Radiant).

## 6. Gleise und Straßen

### 6.1 Prop - locked

**Beschreibung:** locked Sperrt eine Weiche, sodass sie nicht mehr verändert werden kann (Boolean).

### 6.2 Prop - stateCount

**Beschreibung:** stateCount Gibt die Anzahl der Weichenstellungen zurück.

### 6.3 Prop - state

**Beschreibung:** state Setzt die aktive Weichenstellung.

### 6.4 Prop - routeCount

**Beschreibung:** routeCount Gibt die Anzahl der Spuren zurück.

### 6.5 Prop - routes

**Beschreibung:** routes Ermöglicht den Zugriff auf einzelne Spuren anhand des Indexes, beginnend bei 0. Beispiel -- Erste Spur des Gleises "track" aktivieren track.routes[0].active = true

## 7. Gleise und Straßen (Spuren)

### 7.1 Prop - active

**Beschreibung:** active Enthält den Aktivierungszustand einer Spur (Boolean).

### 7.2 Prop - length

**Beschreibung:** length Enthält die Länge der Spur in m.

## 8. Gleiskontakte

### 8.1 Prop - connection

**Beschreibung:** connection Ermöglicht den Zugriff auf das Signal, Gleis oder den Schalter, mit dem der Gleiskontakt verbunden ist (nur lesend).

### 8.2 Prop - autoAcceleration

**Beschreibung:** autoAcceleration Aktiviert oder deaktiviert die automatische Geschwindigkeitsregelung (Beschleunigung).

### 8.3 Prop - autoAccelerationSpeed

**Beschreibung:** autoAccelerationSpeed Enthält die Zielgeschwindigkeit in km/h, auf die ein Fahrzeug bei aktivierter Geschwindigkeitsregelung beschleunigt wird.

### 8.4 Prop - autoDeceleration

**Beschreibung:** autoDeceleration Aktiviert oder deaktiviert die automatische Geschwindigkeitsregelung (Bremsen).

### 8.5 Prop - autoDecelerationSpeed

**Beschreibung:** autoDecelerationSpeed Enthält die Zielgeschwindigkeit in km/h, auf die ein Fahrzeug bei aktivierter Geschwindigkeitsregelung abgebremst wird.

### 8.6 Prop - uncoupling

**Beschreibung:** uncoupling Aktiviert oder deaktiviert die Entkupplungsfunktion des Gleiskontakts.

## 9. Signale

### 9.1 Prop - stateCount

**Beschreibung:** stateCount Gibt die Anzahl der Signalbegriffe zurück.

### 9.2 Prop - state

**Beschreibung:** state Setzt den aktiven Signalbegriff.

### 9.3 Prop - connection

**Beschreibung:** connection Enthält eine Referenz zu dem Objekt, mit dem das Signal verbunden ist, oder nil, wenn keine Verbindung existiert.

## 10. Schalter/Regler

### 10.1 Prop - stateCount

**Beschreibung:** stateCount Gibt die Anzahl der Schalterstellungen zurück.

### 10.2 Prop - state

**Beschreibung:** state Setzt die aktive Schalterstellung.

### 10.3 Prop - value

**Beschreibung:** value Setzt die Reglerposition (zwischen 0 und 1).

### 10.4 Prop - connection

**Beschreibung:** connection Enthält eine Referenz zu einem Ziel-Objekt, das von dem Schalter/Regler gesteuert wird, oder nil, wenn keine Verbindung existiert.

## 11. Schalter (integriert)

### 11.1 Prop - stateCount

**Beschreibung:** stateCount Gibt die Anzahl der Schalterstellungen zurück.

### 11.2 Prop - state

**Beschreibung:** state Setzt die aktive Schalterstellung.

### 11.3 Prop - autoMode

**Beschreibung:** autoMode Steuert den Automatik-Modus des Schalters. Werte nil - Deaktiviert den Automatik-Modus "Day" - Aktiviert den Automatik-Modus "Einschalten bei Tag" "Night" - Aktiviert den Automatik-Modus "Einschalten bei Nacht"

### 11.4 Method - nextState

**Beschreibung:** nextState () Aktiviert die nächste Schalterstellung.

## 12. Fahrzeuge

### 12.1 Prop - maxSpeed

**Beschreibung:** maxSpeed Enthält die Höchstgeschwindigkeit des Fahrzeugs in km/h.

### 12.2 Prop - currentSpeed

**Beschreibung:** currentSpeed Enthält die aktuelle Geschwindigkeit des Fahrzeugs in km/h.

### 12.3 Prop - currentSpeedAbs

**Beschreibung:** currentSpeedAbs Enthält die aktuelle Geschwindigkeit des Fahrzeugs in km/h, relativ zur Fahrtrichtung. Ein negativer Wert dreht die Fahrtrichtung um.

## 12.4 Prop - targetSpeed

**Beschreibung:** targetSpeed Enthält die Zielgeschwindigkeit des Fahrzeugs in km/h.

## 12.5 Prop - targetSpeedAbs

**Beschreibung:** targetSpeedAbs Enthält die Zielgeschwindigkeit des Fahrzeugs in km/h, relativ zur Fahrtrichtung. Ein negativer Wert dreht die Fahrtrichtung um.

## 12.6 Prop - drivingDirection

**Beschreibung:** drivingDirection Enthält die aktuelle Fahrtrichtung des Fahrzeugs (1 für vorwärts, -1 für rückwärts). Bei stehenden Fahrzeugen wird die letzte Fahrtrichtung zurückgegeben, bzw. die Richtung gesetzt, in die das Fahrzeug bei der nächsten Beschleunigung fährt.

## 12.7 Prop - acceleration

**Beschreibung:** acceleration Enthält die positive Beschleunigung des Fahrzeugs in m/s².

## 12.8 Prop - deceleration

**Beschreibung:** deceleration Enthält die negative Beschleunigung (Bremsen) des Fahrzeugs in m/s².

## 12.9 Prop - autoAcceleration

**Beschreibung:** autoAcceleration Aktiviert oder deaktiviert die automatische Beschleunigung.

## 12.10 Prop - autoDeceleration

**Beschreibung:** autoDeceleration Aktiviert oder deaktiviert das automatische Bremsen / sanfte Ankuppeln.

## 12.11 Prop - parkingBrake

**Beschreibung:** parkingBrake Aktiviert oder deaktiviert die Feststellbremse eines nicht angetriebenen Fahrzeugs.

## 12.12 Prop - particlesEnabled

**Beschreibung:** particlesEnabled Enthält den Aktivierungszustand der Dampfpartikel, falls vorhanden.

## 12.13 Prop - target

**Beschreibung:** target Gibt das Ziel des Fahrzeugs an. Je nach Ziel wählt das Fahrzeug bei Abzweigungen automatisch den kürzesten Weg, der zum angegebenen Ziel führt. Das Ziel ist entweder ein einzelner Gleiskontakt, eine Liste von Gleiskontakten, die nacheinander angesteuert werden sollen, oder nil zum Löschen einer Route (bzw. falls keine gültige Route gefunden werden konnte).

## 12.14 Prop - length

**Beschreibung:** length Enthält die Länge des Fahrzeugs (Abstand beider Kupplungen) in m.

## 12.15 Prop - engine

**Beschreibung:** engine Ermöglicht den Zugriff auf den Motor eines Fahrzeugs, falls dieses angetrieben ist. Beispiel -- Motor aktivieren vehicle.engine.active = true

## 12.16 Prop - couplers

**Beschreibung:** couplers Ermöglicht den Zugriff auf die Kupplungen des Fahrzeugs mit dem Index 0 (vordere Kupplung) und Index 1 (hintere Kupplung). Beispiel -- Hintere Kupplung deaktivieren vehicle.couplers[1].enabled = false

## 12.17 Method - isLocatedOn

**Beschreibung:** isLocatedOn (object) Prüft, ob ein Fahrzeug auf einem Gleis/Straße oder einem Gleiskontakt steht. Parameter object - Gleis/Straße bzw. Gleiskontakt, auf den geprüft werden soll. Rückgabewerte Gibt true zurück, falls das Fahrzeug auf dem Gleis/Straße bzw. dem Gleiskontakt steht, ansonsten false.

## 12.18 Method - isHeadingToward

**Beschreibung:** isHeadingToward (object) Prüft, ob ein Fahrzeug in Richtung eines Gleises/Gleiskontakts fährt. Bei einem Fahrzeugverbund (z.B. Zug) wird ausgehend vom vordersten Fahrzeug in Fahrtrichtung geprüft. Parameter object - Gleis/Straße bzw. Gleiskontakt, auf den geprüft werden soll. Rückgabewerte Gibt true zurück, falls das Fahrzeug in Richtung des Gleises/Gleiskontakts fährt, ansonsten false.

## 12.19 Method - hasEngine

**Beschreibung:** hasEngine () Gibt an, ob es sich bei dem Fahrzeug um ein angetriebenes Rollmaterial oder einen Wagen handelt. Rückgabewerte Gibt true zurück, wenn das Fahrzeug einen eigenen "Motor" besitzt, ansonsten false.

# 13. Fahrzeuge (Antrieb)

### 13.1 Prop - active

**Beschreibung:** active Gibt an, ob ein vorhandener Antrieb ein- oder ausgeschaltet ist.

## 14. Fahrzeuge (Kupplungen)

### 14.1 Prop - vehicle

**Beschreibung:** vehicle Enthält eine Referenz zum Fahrzeug, zu dem die Kupplung gehört.

### 14.2 Prop - enabled

**Beschreibung:** enabled Enthält den Aktivierungszustand einer Kupplung.

### 14.3 Prop - connectedCoupler

**Beschreibung:** connectedCoupler Enthält eine Referenz zur Kupplung eines angedockten Fahrzeugs oder nil, falls die Kupplung frei ist.

## 15. Virtuelle Depots

### 15.1 Prop - count

**Beschreibung:** count Gibt die Anzahl der Züge/Fahrzeuge im Depot an.

### 15.2 Prop - entries

**Beschreibung:** entries Ermöglicht den Zugriff auf einzelne Depoteinträge anhand des Indexes, beginnend bei 0. Beispiel -- Name des ersten Zuges im Depot ermitteln name = depot.entries[0].name

### 15.3 Method - add

**Beschreibung:** add (vehicle) Fügt einen Zug/Fahrzeug zum Depot hinzu. Parameter vehicle - Fahrzeug, das zum Depot hinzugefügt werden soll. Ist das Fahrzeug Teil eines Zuges, wird der gesamte Zug zum Depot hinzugefügt.

### 15.4 Method - release

**Beschreibung:** release (index, [target]) Entlässt einen Zug/Fahrzeug aus dem Depot. Parameter index - Gibt die Nummer des Zuges an, der entlassen werden soll. Der erste Zug im Depot beginnt bei 0. target - Gibt optional ein virtuelles Depot an, aus dem der Zug ausfahren soll, falls der Zug nicht aus dem Ursprungsdepot ausfahren soll.

### 15.5 Method - getEntriesByName

**Beschreibung:** getEntriesByName (name) Ermittelt alle Depot-Einträge anhand eines Namens. Parameter name - Name der Einträge, die zurückgeliefert werden sollen. Rückgabewerte Gibt eine Liste aller Indizes der Einträge mit dem angegebenen Namen zurück.

### 15.6 Method - getEntriesByKeyword

**Beschreibung:** getEntriesByKeyword (name) Ermittelt alle Depot-Einträge anhand eines Schlagworts. Parameter name - Name des Schlagworts. Rückgabewerte Gibt eine Liste aller Indizes der Einträge mit dem angegebenen Schlagwort zurück.

## 16. Virtuelle Depots (Einträge)

### 16.1 Prop - name

**Beschreibung:** name Gibt den Namen des Zuges zurück.

### 16.2 Prop - speed

**Beschreibung:** speed Gibt die Geschwindigkeit des Zuges zurück, mit der er das Depot verlassen wird.

### 16.3 Prop - variables

**Beschreibung:** variables Ermöglicht den Zugriff auf die Variablen des Zuges (Triebfahrzeug). Beispiel -- Variable hinzufügen/bearbeiten depot.entries[0].variables["Zähler"] = 1 -- Variable löschen depot.entries[0].variables["Zähler"] = nil

## 17. Portale

### 17.1 Prop - connection

**Beschreibung:** connection Enthält eine Referenz zum verbundenen Portal, oder nil, wenn keine Verbindung existiert.

## 18. Kräne

### 18.1 Method - moveTo

**Beschreibung:** moveTo (target) Bewegt einen Kran zu einem neuen Ziel. Parameter target - Zielobjekt, zu dem sich der Kran bewegen soll. Transportiert der Kran noch keinen Gegenstand, bewegt sich der Kran zu dem Ziel und hebt es auf. Wird ein Gegenstand transportiert, setzt der Kran diesen auf dem Ziel ab. Ist target = nil, wird die Verbindung zum aktuellen Gegenstand aufgehoben.

### 18.2 Method - reset

**Beschreibung:** reset () Löst die Verbindung zu einem aktuell transportierten Gegenstand und fährt in die Ausgangsposition zurück.

### 18.3 Prop - goods

**Beschreibung:** goods Gibt das Transportgut zurück, das der Kran aktuell transportiert, oder nil, falls kein Transport erfolgt.

## 19. Kameras

### 19.1 Prop - target

**Beschreibung:** target Enthält das Objekt, was die Kamera aktuell verfolgt, oder nil, wenn keine Verfolgung aktiv ist.

### 19.2 Prop - fov

**Beschreibung:** fov Gibt den Blickwinkel des Sichtfeldes der Kamera an, in Grad.

## 20. Animationen

### 20.1 Prop - position

**Beschreibung:** position Gibt die aktuelle Position der Animation zwischen Anfang (0) und Ende (1) zurück.

### 20.2 Method - play

**Beschreibung:** play ([position, [direction, [min, [max, [speed]]]]]) Spielt die Animation ab. Parameter position - Spielt die Animation vom Anfang (0) oder vom Ende (1) ab. Der Standardwert ist -1, in diesem Fall wird die Animation von der aktuellen Position abgespielt. direction - Spielt die Animation vorwärts (1) oder rückwärts (-1) ab. Der Standardwert ist 0, in diesem Fall wird die Animation mit der aktuellen Richtung abgespielt. min - Gibt die minimale Position an, oder -1, falls die Untergrenze nicht geändert werden soll. Eine Animation wird immer nur zwischen der minimalen und maximalen Position abgespielt. max - Gibt die maximale Position an, oder -1, falls die Obergrenze nicht geändert werden soll. Eine Animation wird immer nur zwischen der minimalen und maximalen Position abgespielt. speed - Gibt die Abspielgeschwindigkeit an, oder 0, falls die aktuelle Geschwindigkeit nicht geändert werden soll.

### 20.3 Method - stop

**Beschreibung:** stop ([position]) Stoppt die Animation. Parameter position - Stoppt die Animation an einer bestimmten Position (zwischen 0 = Anfang und 1 = Ende). Der Standardwert ist -1, in diesem Fall wird die Animation an der aktuellen Position gestoppt.

## 21. Geräusche

### 21.1 Method - play

**Beschreibung:** play () Spielt das Geräusch ab.

### 21.2 Method - stop

**Beschreibung:** stop () Stoppt das Geräusch.

## 22. Beschriftungen

### 22.1 Prop - text

**Beschreibung:** text Enthält den Text der Beschriftung.

### 22.2 Prop - textAsNumber

**Beschreibung:** textAsNumber Enthält den Text der Beschriftung, interpretiert als Zahl. Hinweis textAsNumber muss dann verwendet werden, wenn die Beschriftung Zahlen enthält und mit diesen gerechnet werden soll, da Lua strikt zwischen Zeichenketten und Zahlen unterscheidet.

## 23. Textfelder

### 23.1 Prop - text

**Beschreibung:** text Enthält den Text des Textfeldes.

### 23.2 Prop - textAsNumber

**Beschreibung:** textAsNumber Enthält den Text des Textfeldes, interpretiert als Zahl. Hinweis textAsNumber muss dann verwendet werden, wenn das Textfeld Zahlen enthält und mit diesen gerechnet werden soll, da Lua strikt zwischen Zeichenketten und Zahlen unterscheidet.

## 24. Partikeleffekte

### 24.1 Prop - active

**Beschreibung:** active Enthält den Aktivierungszustand des dynamischen Partikeleffekts.

## 25. Fahrstraßen

### 25.1 Prop - name

**Beschreibung:** name Gibt den Namen der Fahrstraße an.

### 25.2 Prop - active

**Beschreibung:** active Gibt an, ob die Fahrstraße aktiv oder inaktiv ist.

### 25.3 Prop - state

**Beschreibung:** state Enthält den detaillierten Zustand der Fahrstraße. Mögliche Werte 0 - Fahrstraße ist inaktiv. 1 - Fahrstraße wurde angefordert, wird allerdings von einer anderen Fahrstraße blockiert. 2 - Fahrstraße wird aktiviert und wartet auf Weichen/Signale, die die angegebenen Stellungen einnehmen. 3 - Fahrstraße ist aktiv und gesperrt.

### 25.4 Prop - autoActivate

**Beschreibung:** autoActivate Gibt an, ob eine Fahrstraße automatisch erneut angefordert werden soll, sobald sie wieder freigegeben wird. Diese Eigenschaft eignet sich z.B. für Zentralblöcke, bei denen eine noch aktive Fahrstraße bereits von einem weiteren Zug angefordert wird. Die Anforderung wird automatisch dann bearbeitet, sobald der vorhergehende Zug die Fahrstraße wieder freigibt.

### 25.5 Prop - waypoints

**Beschreibung:** waypoints Ermöglicht den Zugriff auf die Wegpunkte einer Fahrstraße als Liste von Gleiskontakten.

### 25.6 Prop - tracks

**Beschreibung:** tracks Ermöglicht den Zugriff auf die Gleise einer Fahrstraße als Liste.

### 25.7 Prop - variables

**Beschreibung:** variables Ermöglicht den Zugriff auf die Variablen einer Fahrstraße. Beispiel -- Variable hinzufügen/bearbeiten route.variables["Zähler"] = 1 -- Variable löschen route.variables["Zähler"] = nil

### 25.8 Method - canActivate

**Beschreibung:** canActivate () Prüft, ob die Fahrstraße frei ist und sofort ohne Blockade aktiviert werden kann (true).

## 26. Zeitwerte

### 26.1 Prop - value

**Beschreibung:** value Gibt die Zeit als Gleitkommazahl zwischen 0 und 1 zurück.

### 26.2 Prop - hour

**Beschreibung:** hour Gibt die Stunde als Zahl eines Zeitwertes zurück.

### 26.3 Prop - min

**Beschreibung:** min Gibt die Minute als Zahl eines Zeitwertes zurück.

## 27. Ereignisse/Module

### 27.1 Prop - name

**Beschreibung:** name Enthält den Namen des Ereignisses/Moduls.

### 27.2 Prop - enabled

**Beschreibung:** enabled Enthält den Aktivierungszustand des Ereignisses/Moduls.

## 28. Module

### 28.1 Prop - timers

**Beschreibung:** timers Ermöglicht den Zugriff auf die Timer eines Ereignismoduls. Der Index entspricht dem Timernamen. Beispiel -- Einen Timer mit dem Namen "Taktgeber" starten module.timers["Taktgeber"]:start(2, true)

### 28.2 Prop - variables

**Beschreibung:** variables Ermöglicht den Zugriff auf die Variablen eines Ereignismoduls. Beispiel -- Variable hinzufügen/bearbeiten module.variables["Zähler"] = 1 -- Variable löschen module.variables["Zähler"] = nil

## 29. Ereignisse (Benutzerdefiniert)

### 29.1 Method - invoke

**Beschreibung:** invoke (...) Löst das benutzerdefinierte Ereignis aus mit den übergebenen Parametern. Beispiel local event = $("Benutzerdefiniertes Ereignis") event:invoke(42, "Text")

## 30. Timer

### 30.1 Prop - active

**Beschreibung:** active Gibt an, ob der Timer noch läuft oder bereits abgelaufen ist.

### 30.2 Method - start

**Beschreibung:** start (duration, [restart]) Startet den Timer von vorn. Parameter duration - Dauer in Sekunden. restart - Gibt an, ob der Timer automatisch neustarten soll (true) oder nicht (false). Der Standardwert ist true.

### 30.3 Method - stop

**Beschreibung:** stop () Stoppt den Timer.

## 31. Netzwerk

### 31.1 Method - notify

**Beschreibung:** notify (name, [params]) Sendet eine Benachrichtigung über die externe Steuerschnittstelle an alle verbundenen Netzwerk-Clients. Parameter Name - Individueller Benachrichtigungsname. Params - Optionale Parameter, die mit der Benachrichtigung mitgesendet werden.

## 32. Lua Basisfunktionen

### 32.1 Var - _G

**Beschreibung:** _G Eine Variable, welche die globale Umgebung enthält.

### 32.2 Var - _VERSION

**Beschreibung:** _VERSION Eine globale Variable, welche eine Zeichenkette mit der aktuellen Interpreter-Version enthält. Der aktuelle Inhalt dieser Variablen lautet "Lua 5.3".

### 32.3 Func - assert

**Beschreibung:** assert (v, [message]) Verursacht einen Fehler, wenn der Wert des Arguments v falsch (also nil oder false) ist; andernfalls werden alle Argumente zurückgeliefert. message ist eine Fehlermeldung; wenn nicht angegeben lautet der Standard "assertion failed!"

### 32.4 Func - error

**Beschreibung:** error (message, [level]) Beendet die zuletzt aufgerufene geschützte Funktion und liefert message als Fehlermeldung. Die Funktion error selbst liefert keinen Wert. Für gewöhnlich fügt error am Anfang der Nachricht einige Informationen über die Fehlerposition hinzu. Das level-Argument gibt an, wie

die Fehlerposition ermittelt werden soll. Mit Level 1 (Standard) ist die Fehlerposition dort, von wo aus die error-Funktion aufgerufen wurde. Level 2 gibt den Fehler dort an, von wo aus die Funktion, welche error aufgerufen hat, aufgerufen wurde usw. Das Übergeben eines Level 0 unterbindet das Hinzufügen von Fehlerpositions-Informationen zur Nachricht.

## 32.5 Func - ipairs

**Beschreibung:** ipairs (t) Liefert drei Werte: Eine Iterator-Funktion, die Tabelle t und 0, so dass die Konstruktion for i,v in ipairs(t) do body end über die Paare (1,t[1]), (2,t[2]), ..., bis zum ersten nicht mehr in der Tabelle enthaltenen ganzzahligen Schlüssel iterieren wird.

## 32.6 Func - next

**Beschreibung:** next (table, [index]) Ermöglicht einem Programm, alle Felder einer Tabelle zu durchlaufen. Das erste Argument ist eine Tabelle und das zweite ein Index dieser Tabelle. next liefert den nächsten Index dieser Tabelle und dessen verknüpften Wert. Bei einem Aufruf mit nil als zweitem Argument liefert next einen initialen Index und dessen verknüpften Wert. Erfolgt ein Aufruf mit dem letzten Index oder mit nil in einer leeren Tabelle, liefert next nil. Wird das zweite Argument nicht angegeben, wird dieses als nil interpretiert. Im Speziellen können Sie next(t) verwenden, um zu überprüfen, ob eine Tabelle leer ist. Die Richtung, in welcher die Indizes sortiert werden, ist nicht spezifiziert – auch für numerische Indizes. (Um eine Tabelle in numerischer Sortierung zu traversieren, verwenden Sie das numerische for oder die ipairs-Funktion.) Das Verhalten von next ist nicht spezifiziert, wenn während des Durchlaufs ein Wert zu einem nicht existierenden Tabellenfeld hinzugefügt wird. Sie können jedoch bestehende Felder modifizieren. Im Speziellen können Sie bestehende Felder leeren.

## 32.7 Func - pairs

**Beschreibung:** pairs (t) Liefert drei Werte: Die next-Funktion, die Tabelle t und nil, so dass die Konstruktion for k,v in pairs(t) do body end über alle Schlüssel/Werte-Paare der Tabelle t iterieren wird. Siehe Funktion next für Warnungen zur Modifikation von Tabellen während deren Traversierung.

## 32.8 Func - pcall

**Beschreibung:** pcall (f, arg1, ...) Ruft die Funktion f mit den gegebenen Argumenten im geschützten Modus auf. Das bedeutet, dass ein Fehler in f nicht weitergereicht wird; stattdessen fängt pcall den Fehler und liefert einen Statuscode. Das erste Ergebnis ist der Statuscode (ein Wahrheitswert), welcher true ist, wenn der Aufruf ohne Fehler erfolgte. In so einem Fall liefert pcall zusätzlich alle Ergebnisse des Aufrufs nach diesem ersten Ergebnis. Im Falle eines Fehlers liefert pcall false und die Fehlermeldung.

## 32.9 Func - print

**Beschreibung:** print (...) Nimmt eine beliebige Anzahl Argumente entgegen und gibt deren Werte unter Zuhilfenahme der tostring-Funktion in der Ereignis-Protokollierung aus. print ist nicht für formatierte

Ausgaben gedacht, sondern lediglich als schnelle Möglichkeit, Werte auszugeben – typischerweise zum Debugging. Für formatierte Ausgaben verwenden Sie string.format.

### 32.10 Func - select

**Beschreibung:** select (index, ...) Wenn index eine Zahl ist, werden alle Argumente nach dem Argument index geliefert. Andernfalls muss index die Zeichenkette "#" sein, damit select die Gesamtzahl der zusätzlich übergebenen Argumente liefert.

### 32.11 Func - tonumber

**Beschreibung:** tonumber (e, [base]) Versucht das Argument zu einer Zahl zu konvertieren. Wenn das Argument bereits eine Zahl oder eine zu einer Zahl konvertierbaren Zeichenkette ist, liefert tonumber diese Zahl; andernfalls wird nil geliefert. Ein optionales Argument gibt die anzunehmende Basis der Zahl an. Die Basis kann eine Ganzzahl von 2 bis 36 (inkl.) sein. Bei Basen über 10, steht der Buchstabe 'A' (sowohl in Groß- als auch Kleinschreibung) für 10, 'B' für 11 usf. bis 'Z' für 35. Bei der Basis 10 (Standard) können die Zahlen einen Dezimalteil und auch einen optionalen Exponentialteil haben. Bei anderen Basen sind lediglich natürliche Zahlen erlaubt.

### 32.12 Func - tostring

**Beschreibung:** tostring (e) Nimmt ein Argument beliebigen Typs entgegen und konvertiert dieses in eine passende Zeichenkette. Für eine umfassende Kontrolle darüber, wie Zahlen konvertiert werden, verwenden Sie string.format.

### 32.13 Func - type

**Beschreibung:** type (v) Liefert den Wert des einzigen Arguments als Zeichenkette. Die möglichen Ergebnisse dieser Funktion sind "nil" (eine Zeichenkette, nicht den Wert nil), "number", "string", "boolean", "table", "object", "function", "thread" oder "userdata".

### 32.14 Func - xpcall

**Beschreibung:** xpcall (f, msgh, [arg1, ...]) Diese Funktion ist ähnlich wie pcall, mit dem Unterschied, dass Sie einen neuen Error-Handler setzen können.

## 33. Lua Bibliotheken

### 33.1 Module - math

**Beschreibung:** math Diese Bibliothek enthält grundlegende mathematische Funktionen. Alle Funktionen und Konstanten sind in der Tabelle math enthalten.

### 33.2 Module - string

**Beschreibung:** string Diese Bibliothek stellt generische Funktion zur Zeichenkettenverarbeitung zur Verfügung, unter anderem das Suchen und Extrahieren von Teil-Zeichenketten und die Mustersuche. Bei der Indizierung von Zeichenketten unter Lua befindet sich das erste Zeichen an Position 1. Indizies können negativ sein und werden rückwärts interpretiert, also vom Ende der Zeichenkette an. Somit befindet sich das letzte Zeichen an Position -1 usw. Die Bibliothek stellt alle Funktionen in der Tabelle string zur Verfügung. Sie können die Zeichenketten-Funktionen in objektorientierem Stil verwenden. string.byte(s, i) kann beispielsweise als s:byte(i) geschrieben werden. Die Zeichenketten-Bibliothek geht von einer Kodierung mit einem Byte pro Zeichen aus.

### 33.3 Module - utf8

**Beschreibung:** utf8 Diese Bibliothek bietet grundlegende Unterstützung für die UTF-8-Kodierung. Sie stellt alle ihre Funktionen innerhalb der Tabelle utf8 bereit. Diese Bibliothek bietet keine andere Unterstützung für Unicode als die Handhabung der Kodierung. Jede Operation, die die Bedeutung eines Zeichens benötigt, wie z. B. die Klassifizierung von Zeichen, liegt außerhalb ihres Anwendungsbereichs. Sofern nicht anders angegeben, gehen alle Funktionen, die eine Byte-Position als Parameter erwarten, davon aus, dass die angegebene Position entweder der Beginn einer Byte-Sequenz oder eins plus die Länge der betreffenden Zeichenkette ist. Wie in der String-Bibliothek zählen negative Indizes vom Ende des Strings an.

### 33.4 Module - table

**Beschreibung:** table Diese Bibliothek enthält Funktionen zur Manipulation von Tabellen. Alle Funktionen sind in der Tabelle table enthalten.

## 34. Lua mathematische Funktionen

### 34.1 Class var - huge

**Beschreibung:** huge Der Wert HUGE_VAL, welcher größer oder gleich jedem anderen Zahlenwert ist.

### 34.2 Class var - maxinteger

**Beschreibung:** maxinteger Eine Zahl mit dem größten Wert eines Integers.

### 34.3 Class var - mininteger

**Beschreibung:** mininteger Eine Zahl mit dem niedrigsten Wert eines Integers.

### 34.4 Class var - pi

**Beschreibung:** pi Der Wert von Pi.

## 34.5 Class func - abs

**Beschreibung:** abs (x) Liefert den absoluten Betrag von x.

## 34.6 Class func - acos

**Beschreibung:** acos (x) Liefert den Arkuskosinus von x (in Radiant).

## 34.7 Class func - asin

**Beschreibung:** asin (x) Liefert den Arkussinus von x (in Radiant).

## 34.8 Class func - atan

**Beschreibung:** atan (y, [x]) Liefert den Arkustangens von y/x (in Radiant), aber benutzt die Vorzeichen beider Parameter, um den korrekten Quadranten des Ergebnisses zu ermitteln. (Der Fall, dass x gleich 0 ist, wird ebenfalls korrekt behandelt.) Der Standard-Wert von x ist 1, sodass der Aufruf math.atan(y) den Arkustangens von y zurückliefert.

## 34.9 Class func - ceil

**Beschreibung:** ceil (x) Liefert die kleinste Ganzzahl größer oder gleich x.

## 34.10 Class func - cos

**Beschreibung:** cos (x) Liefert den Kosinus von x (in Radiant).

## 34.11 Class func - deg

**Beschreibung:** deg (x) Liefert den Winkel x (in Radiant) in Grad.

## 34.12 Class func - exp

**Beschreibung:** exp (x) Liefert den Wert e hoch x.

## 34.13 Class func - floor

**Beschreibung:** floor (x) Liefert die größte Ganzzahl kleiner oder gleich x.

## 34.14 Class func - fmod

**Beschreibung:** fmod (x, y) Liefert den Rest der Division x durch y, welche den Quotienten auf 0 rundet.

## 34.15 Class func - log

**Beschreibung:** log(x, [base]) Liefert den Logarithmus von x zur Basis base. Der Standard-Wert von base ist e, sodass math.log(x) den natürlichen Logarithmus liefert.

## 34.16 Class func - max

**Beschreibung:** max (x, ...) Liefert den größten Wert seiner Argumente.

## 34.17 Class func - min

**Beschreibung:** min (x, ...) Liefert den kleinsten Wert seiner Argumente.

## 34.18 Class func - modf

**Beschreibung:** modf (x) Liefert zwei Zahlen: Den ganzzahligen und den gebrochenen Anteil von x.

## 34.19 Class func - rad

**Beschreibung:** rad (x) Liefert den Winkel x (in Grad) in Radiant.

## 34.20 Class func - random

**Beschreibung:** random ([m, [n]]) Wird diese Funktion ohne Argumente aufgerufen, liefert sie eine gleichverteilte, reelle Pseudozufallszahl aus dem Intervall [0, 1]. Wird diese Funktion mit einer Ganzzahl m aufgerufen, liefert math.random eine gleichverteilte, ganzzahlige Pseudozufallszahl aus dem Intervall [1, m]. Wird diese Funktion mit zwei Ganzzahlen m und n aufgerufen, liefert math.random eine gleichverteilte, ganzzahlige Pseudozufallszahl aus dem Intervall [m, n].

## 34.21 Class func - randomseed

**Beschreibung:** randomseed (x) Initialisiert den Pseudozufallszahlen-Generator mit einem spezifischen Startwert: Gleiche Startwerte produzieren gleiche Zahlensequenzen.

## 34.22 Class func - sin

**Beschreibung:** sin (x) Liefert den Sinus von x (in Radiant).

## 34.23 Class func - sqrt

**Beschreibung:** sqrt (x) Liefert die Quadratwurzel von x. (Sie können diesen Wert auch durch den Ausdruck x^0.5 berechnen.)

## 34.24 Class func - tan

**Beschreibung:** tan (x) Liefert den Tangens von x (in Radiant).

### 34.25 Class func - tointeger

**Beschreibung:** tointeger (x) Wenn x zu einer Ganzzahl konvertiert werden kann, wird diese Zahl zurückgeliefert, ansonsten nil.

### 34.26 Class func - type

**Beschreibung:** type (x) Liefert "integer", wenn x eine Ganzzahl ist, "float", wenn x eine reele Zahl ist, oder nil, wenn x keine Zahl ist.

### 34.27 Class func - ult

**Beschreibung:** ult (m, n) Liefert true, wenn der Betrag von m kleiner als der Betrag von n ist

## 35. Lua Zeichenkettenfunktionen

### 35.1 Class func - byte

**Beschreibung:** byte (s, [i, [j]]) Liefert den internen numerischen Code der Zeichen s[i], s[i+1], ..., s[j]. Der Standardwert für i ist 1; der Standardwert für j ist i.

### 35.2 Class func - char

**Beschreibung:** char (...) Erwartet 0 oder mehr Ganzzahlen. Liefert eine Zeichenkette mit einer Länge entsprechend der Anzahl der Argumente, wobei jedes Zeichen den gleichen numerischen Code besitzt, wie das korrespondierende Argument.

### 35.3 Class func - find

**Beschreibung:** find (s, pattern, [init, [plain]])) Sucht nach der ersten Übereinstimmung von pattern in der Zeichenkette s. Wenn eine Übereinstimmung gefunden wird, liefert find die Indizes von s, wo dieses Auftreten beginnt und endet; andernfalls liefert es nil. Ein drittes, optionales Argument init legt fest, wo mit der Suche begonnen werden soll; dessen Standardwert ist 1 und kann negativ sein. Der Wert true als viertes optioales Argument plain schaltet die Mustersuchs-Funktionalität ab, so dass die Funktion eine reine "finde Teil-Zeichenkette"-Operation durchführt, wobei keine Zeichen in pattern als "magisch" angesehen werden. Beachten Sie, dass wenn plain gegeben ist, init ebenfalls gegeben sein muss. Wenn das Muster Treffer sichert, werden diese Werte bei einer erfolgreichen Übereinstimmung nach den zwei Indizes ebenfalls zurückgeliefert.

### 35.4 Class func - format

**Beschreibung:** format (formatstring, ...) Liefert eine formatierte Version seiner variablen Anzahl an Argumenten nach der als erstes Argument angegebenen Vorgabe (welche eine Zeichenkette sein muss). Die Formatierungs-Zeichenkette folgt den selben Regeln wie der printf-Familie der standardmäßigen C-

Funktionen. Der einige Unterschied ist, dass die Optionen *, l, L, n, p und h nicht unterstützt werden und es eine zusätzliche Option q gibt. Die q-Option formatiert eine Zeichenkette in einer passenden Form, um sicher durch den Lua-Interpreter gelesen zu werden: Der Aufruf string.format('%q', 'a string with "quotes" and \n new line') wird beispielsweise folgende Zeichenkette erzeugen: "a string with \"quotes\" and \new line" Die Optionen c, d, E, e, f, g, G, i, o, u, X und x erwarten alle eine Zahl als Argument, wohingegen q und s eine Zeichenkette erwarten.

## 35.5 Class func - gmatch

**Beschreibung:** gmatch (s, pattern) Liefert einen Iterator, welcher bei jedem Aufruf den nächsten Treffer aus pattern in der Zeichenkette s liefert. Falls pattern keine Treffer sichert, wird bei jedem Aufruf die komplette Übereinstimmung erzeugt. Beispielsweise wird die folgende Schleife s = "hello world from Lua" for w in string.gmatch(s, "%a+") do  print(w) end über alle Worte der Zeichenkette s iterieren, wobei pro Zeile eines ausgegeben wird. Das nächste Beispiel sammelt alle key=value-Paare der gegebenen Zeichenkette in einer Tabelle: t = {} s = "from=world, to=Lua" for k, v in string.gmatch(s, "(%w+)=(%w+)") do  t[k] = v end Bei dieser Funktion fungiert ein '^' zu Beginn des Musters nicht als Anker, da dies die Iteration verhindern würde.

## 35.6 Class func - gsub

**Beschreibung:** gsub (s, pattern, repl, [n]) Liefert eine Kopie von s, bei welcher alle (bzw. die ersten n, falls gegeben) Treffer von pattern durch eine Ersetzungs-Zeichenkette, welche durch repl spezifiziert wurde und eine Zeichenkette, Tabelle oder Funktion sein kann, ersetzt wurden. gsub liefert als zweiten Wert auch die Gesamtzahl aufgetretener Treffer. Wenn repl eine Zeichenkette ist, wird dessen Wert zur Ersetzung verwendet. Das Zeichen % dient der Maskierung: Jede Sequenz der Form %n in repl mit n zwischen 1 und 9 steht für den Wert der n-ten gesicherten Teil-Zeichenkette (s. u.). Die Sequenz %0 steht für die komplette Übereinstimmung. Die Sequenz %% steht für ein einzelnes %. Wenn repl eine Tabelle ist, wird die Tabelle mit dem ersten Treffer als Schlüssel für jede Übereinstimmung abgefragt; falls das Muster keine Treffer angibt, wird die komplette Übereinstimmung als Schlüssel verwendet. Wenn repl eine Funktion ist, wird diese für jede auftretende Übereinstimmung mit allen gesicherten Teil-Zeichenkette als Argumente in entsprechender Reihenfolge aufgerufen; falls das Muster keine Treffer angibt, wird die komplette Übereinstimmung als einziges Argument übergeben. Wenn der von einer Tabellenabfrage oder Funktion zurückgelieferte Wert eine Zeichenkette oder Zahl ist, wird dies als Ersetzungs-Zeichenkette verwendet; andernfalls – wenn es false oder nil ist – findet keine Ersetzung statt (d. h. die Original-Übereinstimmung wird in der Zeichenkette belassen). Beispiel: x = string.gsub("hello world", "(%w+)", "%1 %1") --> x="hello hello world world" x = string.gsub("hello world", "%w+", "%0 %0", 1) --> x="hello hello world" x = string.gsub("hello world from Lua", "(%w+)%s*(%w+)", "%2 %1") --> x="world hello Lua from" x = string.gsub("home = $HOME, user = $USER", "%$(%w+)", os.getenv) --> x="home = /home/roberto, user = roberto" x = string.gsub("4+5 = $return 4+5$", "%$(.-)%$", function (s)  return load(s)() end) --> x="4+5 = 9" local t = {name="lua", version="5.3"} x = string.gsub("$name-$version.tar.gz", "%$(%w+)", t) --> x="lua-5.3.tar.gz"

### 35.7 Class func - len

**Beschreibung:** len (s) Erhält eine Zeichenkette und liefert dessen Länge. Die leere Zeichenkette "" hat die Länge 0. Eingebettete Nullen werden gezählt, so dass "a\000bc\000" die Länge 5 hat.

### 35.8 Class func - lower

**Beschreibung:** lower (s) Erhält eine Zeichenkette und liefert eine Kopie dieser Zeichenkette, wobei alle Großbuchstaben zu Kleinbuchstaben geändert werden. Alle anderen Zeichen bleiben ungeändert. Die Definition eines Großbuchstabens hängt von der aktuellen Sprache ab.

### 35.9 Class func - match

**Beschreibung:** match (s, pattern, [init]) Sucht nach der ersten Übereinstimmung von pattern in der Zeichenkette s. Falls eine solche gefunden wird, liefert match die Treffer des Musters; andernfalls liefert es nil. Falls pattern keine Treffer spezifiziert, wird die komplette Übereinstimmung zurückgeliefert. Ein drittes, optionales numerisches Argument init gibt an, wo mit der Suche begonnen werden soll; dessen Standardwert ist 1 und kann negativ sein.

### 35.10 Class func - rep

**Beschreibung:** rep (s, n) Liefert eine Zeichenkette, welche eine Verknüpfung von n Kopien der Zeichenkette s darstellt.

### 35.11 Class func - reverse

**Beschreibung:** reverse (s) Liefert eine Zeichenkette, welche die Zeichenkette s umgekehrt darstellt.

### 35.12 Class func - sub

**Beschreibung:** sub (s, i, [j]) Liefert eine Teil-Zeichenkette von s, welche sich von i bis j erstreckt; i und j können negativ sein. Falls j nicht angegeben wird, wird es als -1 angenommen (was das Gleiche wie die Länge der Zeichenkette ist). Im Speziellen liefert der Aufruf string.sub(s, 1, j) ein Präfix von s der Länge j und string.sub(s, -i) liefert ein Suffix von s mit der Länge i.

### 35.13 Class func - upper

**Beschreibung:** upper (s) Erhält eine Zeichenkette und liefert eine Kopie dieser Zeichenkette, wobei alle Kleinbuchstaben zu Großbuchstaben geändert werden. Alle anderen Zeichen bleiben ungeändert. Die Definition eines Kleinbuchstabens hängt von der aktuellen Sprache ab.

## 36. Lua UTF8-Funktionen

### 36.1 Class func - char

**Beschreibung:** char (...) Empfängt null oder mehr Ganzzahlen, konvertiert jede einzelne in die entsprechende UTF-8-Bytefolge und gibt eine Zeichenkette mit der Verkettung all dieser Sequenzen zurück.

### 36.2 Class var - charpattern

**Beschreibung:** charpattern Das Muster (eine Zeichenkette, keine Funktion) "[\0-\x7F\xC2-\xF4][\x80-\xBF]*", das auf genau eine UTF-8-Bytefolge passt, vorausgesetzt, der Inhalt ist eine gültige UTF-8-Zeichenkette.

### 36.3 Class func - codes

**Beschreibung:** codes (s) Gibt Werte zurück, so dass die Konstruktion for p, c in utf8.codes(s) do body end über alle Zeichen in der Zeichenkette s iteriert, wobei p die Position (in Bytes) und c der Codepunkt jedes Zeichens ist. Trifft er auf eine ungültige Bytefolge, wird ein Fehler ausgegeben.

### 36.4 Class func - codepoint

**Beschreibung:** codepoint (s [, i [, j]]) Gibt die Codepunkte (als Ganzzahlen) aller Zeichen in s zurück, die zwischen den Bytepositionen i und j (beide inklusive) beginnen. Der Standardwert für i ist 1 und für j i. Trifft sie auf eine ungültige Bytefolge, wird ein Fehler ausgegeben.

### 36.5 Class func - len

**Beschreibung:** len (s [, i [, j]]) Gibt die Anzahl der UTF-8-Zeichen in der Zeichenkette s zurück, die zwischen den Positionen i und j (beide inklusive) beginnen. Der Standardwert für i ist 1 und für j -1. Wird eine ungültige Bytefolge gefunden, wird ein false Wert plus die Position des ersten ungültigen Bytes zurückgegeben.

### 36.6 Class func - offset

**Beschreibung:** offset (s, n [, i]) Gibt die Position (in Bytes) zurück, an der die Kodierung des n-ten Zeichens von s (von Position i an gerechnet) beginnt. Ein negatives n erhält Zeichen vor der Position i. Die Vorgabe für i ist 1, wenn n nicht negativ ist, und andernfalls #s + 1, so dass utf8.offset(s, -n) den Offset des n-ten Zeichens vom Ende der Zeichenkette erhält. Befindet sich das angegebene Zeichen weder im Betreff noch direkt nach dessen Ende, gibt die Funktion nil zurück. Als Sonderfall, wenn n gleich 0 ist, gibt die Funktion den Beginn der Kodierung des Zeichens zurück, das das i-te Byte von s enthält. Diese Funktion geht davon aus, dass s eine gültige UTF-8-Zeichenkette ist.

## 37. Lua Tabellenfunktionen

### 37.1 Class func - concat

**Beschreibung:** concat (table, [sep, [i, [j]]]) Bei Übergabe einer Tabelle, deren Elemente alle Zeichenketten oder Zahlen sind, wird table[i]..sep..table[i + 1] ... sep..table[j] zurückgegeben. Der Standardwert für sep ist die leere Zeichenkette, der Standard für i ist 1 und der Standard für j ist die Länge der table. Falls i größer als j ist, wird die leere Zeichenkette zurückgeliefert.

## 37.2 Class func - insert

**Beschreibung:** insert (table, [pos], value) Fügt table das Element value an Position pos hinzu, wobei andere Elemente falls notwendig nach oben verschoben werden, um Platz zu schaffen. Der Standardwert für pos ist n + 1, wobei n die Länge der Tabelle ist, so dass der Aufruf table.insert(t, x) x an das Ende der Tabelle t hinzufügt.

## 37.3 Class func - move

**Beschreibung:** move (a1, f, e, t, [a2]) Verschiebt Elemente von Tabelle a1 zu Tabelle a2, was der folgenden Mehrfachzuweisung entspricht: a2[t],... = a1[f],...,a1[e]. Der Standard für a2 is a1. Der Zielbereich kann mit dem Quellbereich überlappen. Gibt die Ziel-Tabelle a2 zurück.

## 37.4 Class func - pack

**Beschreibung:** pack (...) Liefert eine neue Tabelle die alle Argumente in den Schlüsseln 1, 2 usw. speichert und einem zusätzlichen Feld "n" mit der Anzahl der Argumente. Hinweis, die neue Tabelle ist nicht zwingend eine Sequenz.

## 37.5 Class func - remove

**Beschreibung:** remove (table, [pos]) Entfernt das Element an der Position pos aus table, wobei nachfolgende Elemente verschoben werden, um Lücken zu schließen. Liefert den Wert des entfernten Elements. Der Standardwert für pos ist n, wobei n die Länge der Tabelle ist, so dass der Aufruf table.remove(t) das letzte Element der Tabelle t entfernt.

## 37.6 Class func - sort

**Beschreibung:** sort (table, [comp]) Sortiert die Elemente der Tabelle in-place mit der gegebenen Sortierung von table[1] nach table[n], wobei n die Länge der Tabelle ist. Wenn comp gegeben ist, muss dies eine Funktion sein, welche zwei Tabellenelemente annehmen kann und true liefert, wenn das erste Element kleiner als das zweite ist (so dass not comp(a[i+1],a[i]) nach der Sortierung true wird). Wenn comp nicht angegeben ist, wird stattdessen der Standardoperator < von Lua benutzt. Das Sortierverfahren ist nicht stabil; d.h., dass wenn Elemente von der gegebenen Sortierung als gleich betrachtet werden, deren relative Position zueinander von der Sortierung verändert werden kann.

## 37.7 Class func - unpack

**Beschreibung:** unpack (table, [i, [j]]) Liefert Elemente von table. Diese Funktion entspricht return table[i], table[i+1], ..., table[j] Der Standardwert für i is 1 und für j die Länge der Tabelle.

## 38. Lua Schlüsselwörter

### 38.1 Keyword - and

**Beschreibung:** Keine Beschreibung verfügbar.

### 38.2 Keyword - break

**Beschreibung:** Keine Beschreibung verfügbar.

### 38.3 Keyword - do

**Beschreibung:** Keine Beschreibung verfügbar.

### 38.4 Keyword - else

**Beschreibung:** Keine Beschreibung verfügbar.

### 38.5 Keyword - elseif

**Beschreibung:** Keine Beschreibung verfügbar.

### 38.6 Keyword - end

**Beschreibung:** Keine Beschreibung verfügbar.

### 38.7 Keyword - false

**Beschreibung:** Keine Beschreibung verfügbar.

### 38.8 Keyword - for

**Beschreibung:** Keine Beschreibung verfügbar.

### 38.9 Keyword - function

**Beschreibung:** Keine Beschreibung verfügbar.

### 38.10 Keyword - goto

**Beschreibung:** Keine Beschreibung verfügbar.

## 38.11 Keyword - if

**Beschreibung:** Keine Beschreibung verfügbar.

## 38.12 Keyword - in

**Beschreibung:** Keine Beschreibung verfügbar.

## 38.13 Keyword - local

**Beschreibung:** Keine Beschreibung verfügbar.

## 38.14 Keyword - nil

**Beschreibung:** Keine Beschreibung verfügbar.

## 38.15 Keyword - not

**Beschreibung:** Keine Beschreibung verfügbar.

## 38.16 Keyword - or

**Beschreibung:** Keine Beschreibung verfügbar.

## 38.17 Keyword - repeat

**Beschreibung:** Keine Beschreibung verfügbar.

## 38.18 Keyword - return

**Beschreibung:** Keine Beschreibung verfügbar.

## 38.19 Keyword - then

**Beschreibung:** Keine Beschreibung verfügbar.

## 38.20 Keyword - true

**Beschreibung:** Keine Beschreibung verfügbar.

## 38.21 Keyword - until

**Beschreibung:** Keine Beschreibung verfügbar.

## 38.22 Keyword - while

**Beschreibung:** Keine Beschreibung verfügbar.