

Mojo# - Lenguaje de scripting para Training Manager

Sintaxis

- **Literales:**
 - **Numéricos.**
 - **Booleanos.** true / false.
 - **Texto.** Cualquier texto entre comillas. (“”)
- **Operadores** (Siempre que se use una expresión, debe estar entre paréntesis)
 - Unarios
 - **Formato:** operador [literal / (expresión)]
 - **not** Niega la expresión que le sigue
 - Binarios
 - **Formato:** [literal / (expresión)] operador [literal / (expresión)]
 - **Comparadores:** ==, >=, <=, >, <
 - **Matemáticos:** +, -, *, /
 - **Lógicos:** and, or
- **Variables**
 - **Definición**
var [nombrevariable]. (El punto al final es obligatorio)
 - **Acceso**
[nombrevariable].get
 - **Asignación**
[nombrevariable] = [literal / (expresión)]
- **Control de Flujo**
 - **If:** Salto condicional
if (literal / expresión) Cualquier número de expresiones **endif**
- **Métodos Definidos**
 - Permite ejecutar métodos definidos en unas clases especiales.
 - **Formato:** NombredeClase.NombreDeMetodo(parametros)

API

•Asset

- void **Asset.Enable(string assetName, bool enable)** : *Enable/disable the asset based on 'enable' value.*

•Breakdown

- void **Breakdown.Break(string breakdownId)** : *Breaks something on the vehicle. Values: WheelMisalignment - Puncture - SuspensionFailure - NoFuel- Overheating - BrakeFailure - LightsFailure - GlassBroken - LeftMirrorBroken- RightMirrorBroken - ClutchBroken - EngineFailure - SteeringBroken - FifthWheelBroken- DoorBroken - DirtyGlass - AbsBroken - AsrBroken - EspBroken - FrontBrakeFailure- RearBrakeFailure - SnowPlowHydraulicFailure - TransmissionBroken - BlowoutLeftTire*

- void **Breakdown.Repair(string breakdownId)** : *Repairs something on the vehicle.*
Values: Same as previous method.
- void **Breakdown.BreakAll()** : *Fix the whole vehicle*
- void **Breakdown.FixAll()** : *Fix the whole vehicle*

•Bus

- void **Bus.ResetAll()** : *Resets all Passengers*
- void **Bus.StopRequest(string passengerType)** : *Ask for an stop from inside the bus.*
Values: COMMON - HANDICAPPED
- void **Bus.UploadRequest(string passengerType)** : *Ask for an stop from outside the bus.*
Values: COMMON - HANDICAPPED

•Driver

- void **Driver.SetBreathalyzerLevel(string breathalyzerLevel)** : *Set the selected breathalyzer level. Values: NONE - LOW - MEDIUM - HIGH*
- void **Driver.SetTirednessLevel(string tirednessLevel)** : *Set the selected tiredness level.*
Values: NONE - MEDIUM - HIGH

•Environment

- void **Environment.ChangeWeather(string selectedWeather)** : *Set the selected weather.*
Values: SUNNY - CLOUDY - RAINY - ICY - HAIL - SNOW
- void **Environment.ChangeDayTime(string selectedDayTime)** : *Set the selected day time. Values: DAY - DUSK - NIGHT - DAWN .*
- void **Environment.SetIntensity(string selectedIntensity)** : *Set the selected weather intensity. Values: NONE - LOW - MODERATE - HIGH*
- void **Environment.ModifyAsphalt(string selectedAsphalt)** : *Set the selected asphalt modifier. Values: FOG - AQUAPLANING - OIL – DEGRADED ASPHALT - ICY ADHERENCE*
- void **Environment.ChangeWindDirection(string direction)** : *Set the selected wind direction. Values: NORTH, SOUTH, EAST, WEST*
- void **Environment.ChangeWindIntensity(string intensity)** : *Set the selected wind intensity. Values: NONE - LOW - MODERATE – HIGH*

•Exercise

- void **Exercise.Reboot()** : *Reboot the scripts of the exercise.*

•HelpScreens

- void **HelpScreens.PlayVideo**(string name, number xPos, number yPos, number width, number height) : *Play selected video in a window with the indicated size and position.*
- void **HelpScreens.StopVideo**() : *Hide video window if any.*
- void **HelpScreens.AddResult**(string row name, number value) : *Add a row to the results window.*
- void **HelpScreens.ShowResults**(number xPos, number yPos, number width, number height) : *Show the results window.*
- void **HelpScreens.HideResults**() : *Hide the results window*

•Infractions

- void **Infractions.StartChecking**(string infractionId) : *Starts checking an infraction.*
Values: WrongGearShift, SecurityBelt, Stalled, Wiper, HandBrake, SkipBusStop, BusPark, OpenDoor, Clutch, Collision, TrafficVehicleCollision, AnimalCollision, PedestrianCollision, BikeCollision, DrivingLaneReserved, DrivingInOppositeDirection, BadRetarder, OpenTrailer, BusBadPosition, FifthWheelError, BreakDown, PourSnowDown, DrivingOnSnowWithoutShovel, LaneChange, HittingExpansionJoints, RemoveSnowWithoutEmergencyLights, SpreadSaltWithoutEmergencyLights, ExcessSpeedRemovingSnowToPlow, PourSnowToAdjacentLanes, PourSnowOnVehicles, FreeAcceleration, GearShiftAcceleration, IgnitionWithKickstand, DrivingWithoutSnowWithShovel, PriorityNoRespectedInPedestrianCrossing, PriorityNoRespected, SpeedLimit, TooSlow, Offroad, ConeCollision, LightsInfractionBadSituation, LightsInfractionMandatorySituation, ExcessSpeedRemovingSnowToWedge, RotaryLights, Traction, TransferBox, Differential, Chains, Inclination, RespectCyclist, CollisionWithTrain, DistanceExceededAtBusStop, AggressiveDriving, UnboardHandicappedNotDone, BoardHandicappedNotDone, UnboardPassengersNotDone, BoardPassengersNotDone, ToRunARedLight, InstructorCollision, ToRunAStop, ObstructionToEmergencyVehicles, SettedSpeedLimit, NeutralGearDriving, OperatorsSecurityDistance, Overturn, WrongApproachToCrusher, UnloadWhileMoving, OperatorCollision, DumperBucketCollision, PowerLineCollisionInfraction, NotUseRetarderDownhill, DrivingWithBucketRaised, ParkWithBucketRaised, ToRunAStopWithTrigger, UnloadInWrongPlace, DrivingWithForkRaised, DamagedFurniture, DamagedLoad, WrongGearShiftMX, ToRunARailway, StoppedInRailway, ParkedInRailway

- void **Infractions.StopChecking(string infractionId)** : *Stops checking an infraction. Values: Same as previous method.*
- bool **Infractions.HasBeenCommitted(string infractionId)** : *Returns a value indicating if an infraction that is being checked has been committed. Values: Same as previous method.*
- int **Infractions.Count(string infractionId)** : *Returns the number of times an infraction has been committed since the beginning of the exercise. Values: Same as previous method.*

•Input

- void **Input.WaitAction(string action)** : *Wait until action input is pressed. Values: Continue.*

•Locution

- void **Locution.Play(string locutionToPlay, bool wait)** : *Play the selected locution. If there's any locution playing, it will stop. 'wait' is used to stop script execution until locution is played.*
- void **Locution.PlayOneShot(string locutionToPlay, bool wait)** : *Play the selected locution just once. To play it again you have to do a Locution.Restart(). If there's any locution playing, it will stop.*
- void **Locution.Restart()** : *Allow to play again any locution*
- void **Locution.Stop()** : *Stop the locution being played, if any*

•Logger

- void **Logger.ShowMessage(string message)** : *Show message in the indications window*
- void **Logger.HideMessage()** : *Hide current message in the indications window*

•Objective

- void **Objective.Increase(string objectiveId)** : *Increase the Objective score by one*
- void **Objective.Decrease(string objectiveId)** : *Decrease the Objective score by one*

•Operations

- float **Operations.Distance(string point1, string point2)** : *Distance between two points*
- float **Operations.Clamp(number value, number min, number max)** : *The float result between the min and max values*
- string **Operations.Concatenate(string string1, string string2)** : *Adds two strings*

•Timer

- void **Timer.WaitSeconds(float seconds)** : *Waits indicated seconds before continuing execution*
- void **Timer.FinishExercise()** : *Finishes current exercise*
- float **Timer.Timestamp()** : *Returns float timestamp in milliseconds*

•Traffic

- void **Traffic.ChangeIntensity(string selectedIntensity)** : *Sets the traffic intensity. Values: NONE - LOW - MODERATE – HIGH*
- void **Traffic.ChangeIntensityPct(float intensity)** : *Sets the traffic intensity to the given percentage value (1-100)*
- void **Traffic.Cross(string selectedEntity, string selectedDifficulty)** : *Makes an entity cross in front of the user vehicle. Values for Entity: PEDESTRIAN - ANIMAL - BICYCLE - TORNADO - DOG - HORSE - COW – PIG. Values for Difficulty: EASY - MEDIUM - HARD*
- void **Traffic.Accident(string selectedDifficulty)** : *Spawns an accident in front of the user vehicle. Values: EASY - MEDIUM - HARD*
- void **Traffic.BrokenCar(string selectedDifficulty)** : *Spawns a broken car in front of the user vehicle. Values: EASY - MEDIUM - HARD*
- void **Traffic.Ambulance(string selectedDifficulty)** : *Spawns an ambulance in front of User Vehicle. Values: EASY - MEDIUM - HARD*
- void **Traffic.StartTrafficRoute(string routeName)** : *Starts traffic route*
- void **Traffic.StopTrafficRoute(string routeName)** : *Stops traffic route*
- void **Traffic.TrafficJam()** : *Spawns a traffic jam in front of User Vehicle*
- void **Traffic.ClearTrafficEntities()** : *Clears all spawned traffic entitie*

•Trigger

- void **Trigger.Enable(string triggerName, bool enable)** : *Enable/disable the trigger based on 'enable' value.*

•UserVehicle

- object **UserVehicle.GetValue(string id)** : *Returns the desired value from Vehicle. Possible Values: RPM, Gear, Velocity, FuelConsumption, AverageFuelConsumption, AccumulatedConsumption, Emissions, MeanEmissions, AccumulateEmissions,*

DistanceTraveled, HandBrake, Retarder, ClutchPedal, GasPedal, BrakePedal, Belt, PositionLights, BeamLights, FullBeamLights, FrontFogLights, RearFogLights, LeftBlinkerLights, RightBlinkerLights, WarningLights, Wipers, FrontBrake, RearBrake, UserPosition

- **void UserVehicle.Respawn(string spawnPointName)** : Respawns the vehicle on the selected SpawnPoint
- **void UserVehicle.Enable(bool isEnabled)** : Enable or disable vehicle controls
- **void UserVehicle.ChangeLoadLevel(float loadLevel)** : Sets the load level. Values: 0 - 0.25 - 0.50 - 0.75 - 1

Recomendaciones y advertencias

- Al hacer una comparación de una variable con un literal, poner el literal en la parte izquierda de la comparación.
- Las expresiones deben escribirse entre paréntesis, ya sea en una asignación de una variable, en la parte izquierda o derecha de un operador, como condición de un if, etc. Ej:

```
var x.
```

```
x = 5
```

```
x = UserVehicle.GetValue("RPM")
```

```
x = (x.get + 1)
```

```
if ( 1 > UserVehicle.GetValue("Velocity") )
```

```
    Locution.Play("Stopped", true)
```

```
endif
```

```
if ( ( 50 + 30 ) < UserVehicle.GetValue("Velocity") )
```

```
    Locution.Play("Too fast", true)
```

```
endif
```

- Para lanzar una ruta, no basta con crearla en el editor, hay que darle un nombre y lanzarla desde Mojo# con la instrucción Traffic.StartTrafficRoute("nombreDeRuta").
- Si se va a usar el método OnStay de un trigger, no es recomendable (aunque se permite) utilizar los métodos OnStart y OnExit del mismo trigger para evitar conflictos en el orden de ejecución de las instrucciones.
- El método UserVehicle.GetValue("Velocity") devolverá un valor negativo en caso de que el vehículo vaya marcha atrás.

- El método `Exercise.Reboot()` vuelve a ejecutar el ejercicio desde el principio, pero sólo en lo que a los scripts se refiere. Si por ejemplo queremos que el vehículo vuelva a estar en su posición inicial, tendríamos que hacer un Respawn a esa posición en el script `OnStart`.
- El método `Input.WaitAction("Continue")` se utiliza para que la ejecución de los scripts se detenga hasta que pulsemos la tecla Enter (la del centro de la cruceta en la plataforma). De momento no admite más valores.
- Los ejercicios de Training Manager deben estar en la misma carpeta en la que se guardan por defecto (`Application/Simescar_Data/StreamingAssets/TrainingManager`). Si cargamos un ejercicio que no está en ese directorio, ocurrirá lo siguiente:
 - En el Editor TM:
 - No se cargarán los scripts. Cualquier método (`OnHit`, `OnStay`, `OnEnter`, etc.) aparecerá vacío.
 - Si le damos a Guardar, **todos los scripts se borrarán**, ya que el ejercicio se guarda con su estado actual, es decir, con los scripts vacíos.
 - Cargando del ejercicio en simulación:
 - El ejercicio carga, con los objetos que hayamos puesto, pero los scripts tampoco se cargan, por lo que nada de lo programado en `Mojo#` funciona.