

# ⬆️ <> Wotja 24 Script Engine



The Wotja Script Engine ("WSE") allows programmatic control of WME parameters. It's use is entirely optional but it comes into its own when creating "Adaptive Music".

[Current version](#) | [24 Feature Set](#) | [Standalone WSE Guide](#) | [WSE Guide PDF](#) > [Doc Archive](#)

## New in Wotja 24

- Wotja Script extended for use of:
  - [Generator Rules](#)
  - [Cell Rules](#)
  - [Templates \(incl. in WJURI form\)](#)
  - [WAE Synth & FX Network Presets](#)

Looking for something?: *Try page search* with Cmd+F/Ctrl+F.


### Quick links

- [Scripting Overview](#)
- [Script Editor](#)
- [Mix Script](#)
- [Cell Script](#)
- [Generator Script](#)
- [Scripting Objects](#)
- [Script Function Reference](#)
- [Scripting Cookbook](#)
- [Hyperinstrument](#)

## <> Scripting Overview

Wotja Scripting is aimed at those who want to push the outer limits of what Wotja can do out of the box. Wotja scripts are based on ECMAScript and edited in the [Script Editor](#) and Script Sandbox. Scripts can be used in 3 contexts: Mix Scripts, Cell Scripts, Generator Scripts. We give you lots of examples to get you started.

**Wotja Script language:**

The [ECMAScript](#)  scripting language (ECMAScript E5/E5.1) is widely used in the web, gaming and multimedia worlds. It is very powerful, fast and easy to learn and commonly referred to as JavaScript.

### Wotja Script Editor:

- **Mix Script** - Script saved to the Mix Script parameter:
  - Wotja [Music Edit Mode](#) > [Mix Properties](#) > [Mix Script parameter](#)
- **Cell Script** - Script saved to the Cell Script parameter:
  - Wotja [Music Edit Mode](#) > [Cell Properties](#) > [Cell Script parameter](#)
- **Generator Script** - Script saved to the Generator Script parameter:
  - Wotja [Mix Edit Mode](#) > [Editor Panel \(WME\)](#) > [Generator \(AKA Generator\)](#) > [Generator Script parameter](#)

### Wotja Script Sandbox:

*Tip:* The Sandbox is for you to experiment with Wotja Script. Sandbox scripts are not saved to files and don't access Mix properties.

- Wotja [Mix Mode](#) > Action Menu > Script Sandbox

## Why would you want to use Scripting?

Are you a generative music fan who wants to see what your own custom code can achieve when leveraging a hugely powerful app such as Wotja? Are you code hacker who want to play with music? Are you a Wotja expert who wants to push the outer limits of what Wotja can do out of the box?

With careful use of Wotja's Event Handler Script Functions, you can do a lot of cool things, while turning Wotja into [a custom hyper-instrument](#).

You can, if you wish, create Event Handler Script Functions to run in response to the following events:

- start of playback
- every bar
- when a note is composed by the WME
- end of playback
- in response to external MIDI CC via e.g. keyboard controller
- in response to external MIDI Note on/off via e.g. keyboard controller

Every script can manipulate a large number of things in Wotja in real-time. Some examples: manipulating music rules, playing with mutation, altering patterns; and what have you.

By creating code to respond to external MIDI events, you can in turn manipulate various properties of Wotja, such as auto-chording parameters, rule settings etc.

## User Interface Updates

If you change a property with a script function such as *wmeParameterSet*, the user interface doesn't always update to reflect your call; you'd need to refresh the table that displays your updated parameter, to see the change in the user interface. We hope to be able to improve this behavior at some future point.

If you change a parameter with the user interface, and that parameter is later changed by a script; the latter script call will override your user interface change. Don't get confused by this!

If you have a script which copies a parameter value at start, and then resets it when the mix stops; where you've in between changed the parameter through the user interface; the latter script call will override your user interface change. Don't get confused by this!

## Latency and Timing

The WME composes events a little bit in advance of when you'll hear them. This is to ensure that the notes are composed, in time for them to be picked-up by the WAE; the system also does the audio processing required by the WAE a little ahead of time before you actually hear it. Added together, this means that script code run in response to, say, *onWmeBar* won't always appear precisely synchronized to what you're listening to!

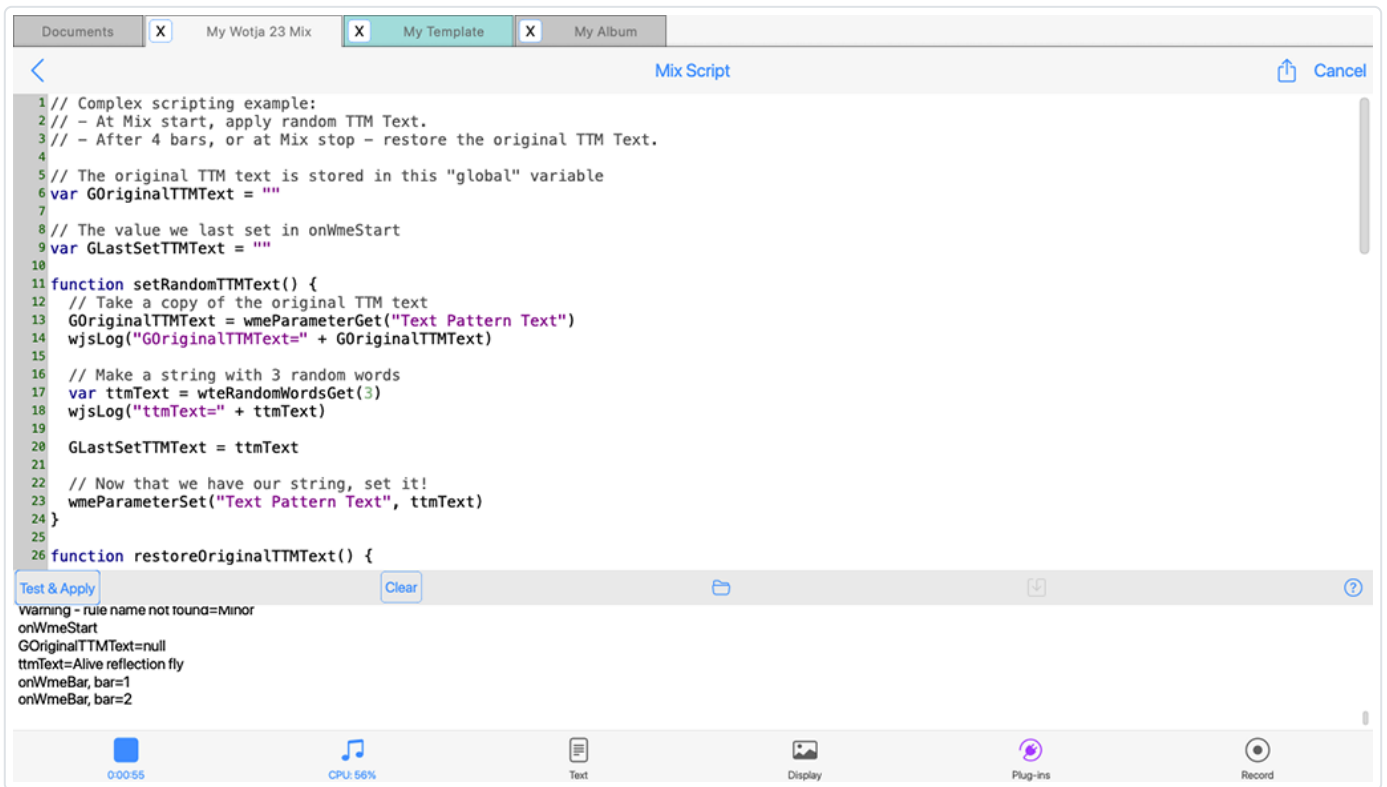
## Infinite Loops

Please be careful not to implement so-called *Infinite Loops* in your script code, as that might cause the Wotja application to crash.

## Recursion (recursive function calls)

Please be careful not to write script code that results in a lot of *Recursion*, as that might cause the Wotja application to crash.

## Script Editor / Sandbox



### Wotja Script Editor Interface

Click image for large

When you decide to edit a *Mix Script* or *Cell Script* or *Generator Script* parameter, or if you select the *Script Sandbox* menu item, you are presented with the *Script Editor* window. It supports syntax-colored code editing and a console window at the bottom lets you capture logs from your scripts.

- Mix/Cell/Generator Script Editor  
In this mode, the top panel allows you to edit Wotja Script Functions associated with your Mix or Cell or Generator.
- Script Sandbox  
In this mode, it is used to create and run scripts that aren't saved, and that don't access Mix properties. It is there for you to experiment with Wotja Script!

## Script Editor: Top Menu Bar

## < Back

Pressing the Back button will close the screen. If you're editing a Generator Script or Cell Script or Mix Script, you'll first be prompted to save or discard any changes you might have made.

## Title

Will be either "Generator/Cell/Mix Script" or "Scripting Sandbox" depending how you have accessed it.

## 📄 Action

Pressing the *Action* button will show the following popup menu:

- *Edit...*: Allows you to edit in a standard text editor with standard copy/paste etc.
- *Clear Script*: Select this to clear the script.
- *Export to Clipboard*: Exports script to clipboard.
- *Import from Clipboard*: Imports script from clipboard.

## Cancel

Pressing the *Cancel* button will discard your changes.

## Script Editor: Top Text Panel

This is where you edit Wotja Script Functions associated with your object.

The panel can contain as many functions as you want, or as few as you want. Leave it blank if you don't want to use any scripting.

## Logging output to the Script console with *wjsLog*

The *wjsLog* function may be used to write text to the [Script Console](#); this is very useful for debugging your scripts!

Here is an example with just one trigger script function:

```
// Just one function!

function onWmeBar(bar) {
    wjsLog("onWmeBar: Bar number=" + bar)
}
```

Here is an example with three trigger script functions:

```
// Three functions!

function onWmeStart() {
    wjsLog("onWmeStart!")
}

function onWmeBar(bar) {
    wjsLog("onWmeBar: Bar number=" + bar)
}

function onWmeStop() {
    wjsLog("onWmeStop!")
}
```

## Script Editor: Bottom toolbar

### Test & Apply (only for Generator/Cell/Mix Script)

Pressing this button will compile any Wotja Script that you type in the large text area at the top of the window, and if the script looks valid, it will send the script to be stored within the Mix. It will take effect immediately (if your Mix is running).

### Run Script (only for Script Sandbox)

Pressing this button will compile any Wotja Script that you type in the large text area at the top of the window, and display any results in the bottom panel. Use *Run Script* to quickly check your script for obvious syntax errors.

Note that any changes you make here aren't saved within the mix.

### Clear Log

Pressing the *Clear Log* button in the bottom area, quickly erases the text in the bottom panel.

### Open

Pressing the Open button displays the Wotja Scripts Window (below), where you can select a Wotja Script file from either built-in examples, or from ones you've saved yourself (which have a

.wotjascript extension). Your saved custom scripts are listed in the top section; you can save your own custom scripts in this section, by using the Save button.

## Wotja Scripts Window

The Wotja Scripts window lets you select from a number of built-in scripts that are provided by Intermorphic. There are simple implementations of each of the Event Triggers functions for Generator Scripts or Cell Scripts or Mix Scripts, and there are a large number of *Cookbook* scripts for both Generator and Mix. The *Cookbook* scripts which are often quite detailed, and which are designed to help you get started with writing your own scripts.

When in this Window, select the item you wish to use in the Script Editor.

The top section lists any Scripts that you might have saved yourself; you can press the "Trash Can" icon to delete your user script.

### Save

Pressing the Save button lets you save the contents of your Script editor window to a document with the .wotjascript extension. You can then easily find and re-open that Script using the Open button (your saved custom scripts are listed in the top section).

### Help

Pressing the *Help* button displays help on Wotja Scripting system!

## Script Editor: Test Results Panel (Script Console)

Any test results are displayed here.

### See Also

- [Scripting Overview](#)
- [Mix - Script](#)
- [Cell - Script](#)
- [Generator - Script](#)
- [Event Handler Script Functions](#)
- [Script Function Reference](#)

## Mix Script

### Mix Script

- [onWmeStart\(\)](#)
  - [onWmeBar\(\)](#)
  - [onWmeStop\(\)](#)
  - [onMIDIInCC\(\)](#)
  - [onMIDIInNote\(\)](#)
- 

Mix Script applies to the whole Mix and is stored in the [Mix Script parameter](#). *Tip:* For organisational purposes it can help if you put all the scripts used by the Mix in Mix Script parameter. That said, you will find that some templates contain generator-specific script code in the [Generator Script parameter](#), and you might choose to put cell-specific script code in the [Cell Script parameter](#).

Mix-related **Event Handler Script Functions** ("*Triggers*", and which all start with **onWME**) get triggered when the Mix is playing and when certain events happen.

Use of these Triggers lets you affect how the Mix plays at certain points (or via detection of certain events) by letting you use various [Scripting Functions](#) to change the [Mix Object](#), [Cell Objects](#), [Generator Objects](#) and [Rule Objects](#).

## [onWmeStart\(\)](#)

### Start

The **onWmeStart** Event Handler Script Function is called once at the start of the Mix playing.

The function has no parameters.

```
function onWmeStart() {  
    wjsLog("Mix start!")  
}
```

## [onWmeBar\(\)](#)

### Bar

The **onWmeBar** Event Handler Script Function is called at the start of every bar while the mix is playing, starting from 1.

The function has one parameter:

- *bar* the bar number, starting from 1.



```
function onWmeBar(bar) {
  wjsLog("Bar number=" + bar)
}
```

 `onWmeStop()`

## Stop

The **onWmeStop** Event Handler Script Function is called once, just as the Mix stops playing.

The function has no parameters.

```
function onWmeStop() {
  wjsLog("Mix stop!")
}
```

 `onMIDIInCC()`

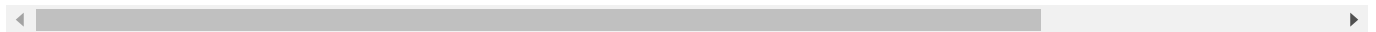
## MIDI In CC

The **onMIDIInCC** Event Handler Script Function is called whenever a MIDI CC event is received by the MIDI Input device.

The function has three parameters:

- *channel* the MIDI Channel, from 0 to 15 (MIDI channel is actually displayed in the WME Network screen as 1 to 16)
- *cc* the CC event identifier, from 0 to 127.
- *value* the value of the CC event, from 0 to 127.

```
function onMIDIInCC(channel, cc, value) {
  wjsLog("Mix onMIDIInCC channel" + channel + ", " + cc
}
```



## onMIDIInNote()

### MIDI In Note

The **onMIDIInNote** Event Handler Script Event Handler Script Function is called whenever a MIDI Note On or Off event is received by the MIDI Input device.

The function has four parameters:

- *channel* the MIDI Channel, from 0 to 15 (MIDI channel is actually displayed in the WME Network screen as 1 to 16)
- *noteon* a value indicating if this is a note on event - true means note on, false means note off.
- *pitch* the MIDI pitch of the note event, from 0 to 127.
- *velocity* the velocity of the note event, from 0 to 127.

```
function onMIDIInNote(channel, noteon, pitch, velocity)
    wjsLog("Mix MIDI in note=" + noteon + ", " + channel
}
```

### See Also

- [Overview](#)
- [Cell Script](#)
- [Generator Script](#)
- [Scripting Overview](#)
- [Script Function Reference](#)
- [Wotja as a Hyperinstrument](#)

## Cell Script

### Cell Script

- [onWmeStart\(\)](#)
- [onWmeBar\(\)](#)
- [onWmeStop\(\)](#)
- [onMIDIInCC\(\)](#)
- [onMIDIInNote\(\)](#)

---

Cell Script applies to a specific Cell and is stored in the [Cell Script parameter](#). *Tip:* For organisational purposes it can help if you put all the scripts used by the Mix in Mix Script parameter. That said, you will find that some templates contain generator-specific script code in the [Generator Script parameter](#), and you might choose to put cell-specific script code in the [Cell Script parameter](#).

Cell-related **Event Handler Script Functions** ("*Triggers*", and which all start with **onWME**) get triggered when the Cell is playing and when certain events happen.

Use of these Triggers lets you affect how the Cell plays at certain points (or via detection of certain events) by letting you use various [Scripting Functions](#) to change the [Mix Object](#), [Cell Objects](#), [Generator Objects](#) and [Rule Objects](#).

 **onWmeStart()**

## Start

The **onWmeStart** Event Handler Script Function is called once at the start of the Mix's owning cell, when that cell starts playing.

The function has no parameters.

```
function onWmeStart() {
  wjsLog("Mix start!")
}
```

 **onWmeBar()**

## Bar

The **onWmeBar** Event Handler Script Function is called at the start of every bar while the mix is playing, starting from 1.

The function has one parameter:

- *bar* the bar number, starting from 1.

```
function onWmeBar(bar) {
  wjsLog("Bar number=" + bar)
```

```
}
```

## `onWmeStop()`

### Stop

The **onWmeStop** Event Handler Script Function is called once at the end of the Mix's owning cell, just as that Cell stops playing.

The function has no parameters.

```
function onWmeStop() {  
    wjsLog("Mix stop!")  
}
```

## `onMIDIInCC()`

### MIDI In CC

The **onMIDIInCC** Event Handler Script Function is called whenever a MIDI CC event is received by the MIDI Input device.

The function has three parameters:

- *channel* the MIDI Channel, from 0 to 15 (MIDI channel is actually displayed in the WME Network screen as 1 to 16)
- *cc* the CC event identifier, from 0 to 127.
- *value* the value of the CC event, from 0 to 127.

```
function onMIDIInCC(channel, cc, value) {  
    wjsLog("Mix onMIDIInCC channel" + channel + ", " + cc  
}
```



## `onMIDIInNote()`

### MIDI In Note

The **onMIDIInNote** Event Handler Script Event Handler Script Function is called whenever a MIDI Note On or Off event is received by the MIDI Input device.

The function has four parameters:

- *channel* the MIDI Channel, from 0 to 15 (MIDI channel is actually displayed in the WME Network screen as 1 to 16)
- *noteon* a value indicating if this is a note on event - true means note on, false means note off.
- *pitch* the MIDI pitch of the note event, from 0 to 127.
- *velocity* the velocity of the note event, from 0 to 127.

```
function onMIDIInNote(channel, noteon, pitch, velocity)
  wjsLog("Mix MIDI in note=" + noteon + ", " + channel
}
```

## See Also

- [Overview](#)
- [Mix Script](#)
- [Generator Script](#)
- [Script Function Reference](#)
- [Wotja as a Hyperinstrument](#)

## <> Generator Script

### Generator Script

- [onWmeStart\(\)](#)
- [onWmeBar\(\)](#)
- [onWmeStop\(\)](#)
- [onWmeGeneratorComposed\(\)](#)
- [onWmeGeneratorUserController\(\)](#)
- [onMIDIInCC-generator\(\)](#)
- [onMIDIInNote-generator\(\)](#)

---

Generator Script applies to a specific Generator and is stored in the [Generator Script parameter](#).

*Tip:* For organisational purposes it can help if you put all the scripts used by the Mix in Mix Script

parameter. That said, you will find that some templates contain generator-specific script code in the [Generator Script parameter](#), and you might choose to put cell-specific script code in the [Cell Script parameter](#).

Generator-related **Event Handler Script Functions** (*Triggers*), and which all start with **onWME**) get triggered when the Mix is playing and when certain events happen.

Use of these Triggers lets you affect how the Mix plays at certain points (or via detection of certain events) by letting you use various [Scripting Functions](#) to change the [Mix Object](#), [Cell Objects](#), [Generator Objects](#) and [Rule Objects](#).

## **onWmeStart()**

### **Start**

The **onWmeStart** Event Handler Script Function is called once at the start of the Generator's owning cell, when that cell starts playing.

The function has no parameters.

```
function onWmeStart() {
  wjsLog("Generator start!")
}
```

## **onWmeBar()**

### **Bar**

The **onWmeBar** Event Handler Script Function is called at the start of every bar while the mix is playing in its cell, starting from 1.

The function has one parameter:

- *bar* the bar number, starting from 1.

```
function onWmeBar(bar) {
  wjsLog("Generator Bar number=" + bar)
}
```

## `onWmeStop()`

### Stop

The **onWmeStop** Event Handler Script Function is called once at the end of the Generator's owning cell, just as that Cell stops playing.

The function has no parameters.

```
function onWmeStop() {  
    wjsLog("Mix stop!")  
}
```

## `onWmeGeneratorComposed()`

### Composed

The **onWmeGeneratorComposed** Event Handler Script Function is called when the Generator composes a note. Use this to emit MIDI CC events and what have you.

The function has five parameters:

- *generatorIndex* the generator index, starting from 0.
- *noteon* a value indicating if this is a note on event - true means note on, false means note off.
- *channel* the MIDI Channel, from 0 to 15 (MIDI channel is actually displayed in the WME Network screen as 1 to 16)
- *pitch* the MIDI pitch of the composed note, from 0 to 127.
- *velocity* the MIDI velocity of the composed note, from 1 to 127.

```
function onWmeGeneratorComposed(generatorIndex, noteon, c  
    wjsLog("Generator Composed index=" + generatorIndex + "  
}
```

## `onWmeGeneratorUserController()`

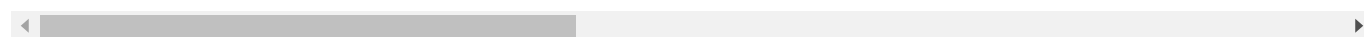
### User Controller Changed

The **onWmeGeneratorUserController** Event Handler Script Function is called when either of the Generator's User Controllers output value changes.

The function has three parameters:

- *generatorIndex* the generator index, starting from 0.
- *controllerIndex* the zero-based index of the controller; 0 means User Controller 1, 1 means User Controller 2.
- *value* the controller's new value, from 0 to 127.

```
function onWmeGeneratorUserController(generatorIndex, controllerIndex, value) {
    wjsLog("onWmeGeneratorUserController, generatorIndex=" + generatorIndex + ", controllerIndex=" + controllerIndex + ", value=" + value);
}
```



## onMIDIInCC()

### MIDI In CC

The **onMIDIInCC** Event Handler Script Function is called whenever a MIDI CC event is received by the MIDI Input device.

The function has three parameters:

- *channel* the MIDI Channel, from 0 to 15 (MIDI channel is actually displayed in the WME Network screen as 1 to 16)
- *cc* the CC event identifier, from 0 to 127.
- *value* the value of the CC event, from 0 to 127.

Note that your script will *only* respond to MIDI In CC Events that target the MIDI Channel that your generator is attached to.

```
function onMIDIInCC(channel, cc, value) {
    wjsLog("Generator onMIDIInCC channel=" + channel + ", cc=" + cc + ", value=" + value);
}
```



## onMIDIInNote()



## MIDI In Note

The **onMIDIInNote** Event Handler Script Event Handler Script Function is called whenever a MIDI Note On or Off event is received by the MIDI Input device.

The function has four parameters:

- *channel* the MIDI Channel, from 0 to 15 (MIDI channel is actually displayed in the WME Network screen as 1 to 16)
- *noteon* a value indicating if this is a note on event - true means note on, false means note off.
- *pitch* the MIDI pitch of the note event, from 0 to 127.
- *velocity* the velocity of the note event, from 0 to 127.

Note that your script will *only* respond to MIDI In Note Events that target the MIDI Channel that your generator is attached to.

```
function onMIDIInNote(channel, noteon, pitch, velocity)
    wjsLog("Generator onMIDIInNote channel=" + channel + "
}
```

## See Also

- [Overview](#)
- [Mix Script](#)
- [Cell Script](#)
- [Script Function Reference](#)
- [Wotja as a Hyperinstrument](#)

## <> Scripting Objects

### Objects and Parameters

The following table lists the various Objects ([Mix](#), [Cell](#), [Generator](#) and [Rule](#)) and their associated parameters together with how those objects/parameters are available to Wotja Scripts, showing: the object name; the parameter group / view (where shown to a user); the displayed name (where

shown to a user); the parameter name (as supplied to functions and which is always unique for a given object); and the range of legal values that you may supply.

## <> Object: "Mix"

Parameter Group / View	Displayed Name	Script Parameter Name	Notes
Mix Rules	Scale Rules	"Scale Rules"	This parameter contains the name of the Scale rule that the Mix is configured to use. The specific rule will therefore be used if a generator/cell is configured to use the "Mix" Scale Rule.
	Harmony Rules	"Harmony Rules"	This parameter contains the name of the Harmony rule that the Mix is configured to use. The specific rule will therefore be used if a generator/cell is configured to use the "Mix" Harmony Rule.
	Next Note Rules	"Next Note Rules"	This parameter contains the name of the Next Note rule that the Mix is configured to use. The specific rule will therefore be used if a generator/cell is configured to use the "Mix" Next Note Rule.
	Rhythm Rules	"Rhythm Rules"	This parameter contains the name of the Rhythm rule that the Mix is configured to use. The specific rule will therefore be used if a generator/cell is configured to use the "Mix" Rhythm Rule.
Mix	Mix Root	"Mix Root"	"A", "A#", "Ab" etc.
	Tempo	"Tempo"	Value in range 1, 400

## <> Object: "Cell"

Parameter Group / View	Displayed Name	Script Parameter Name	Notes
Cell Generative Rules	Scale Rules	"Scale Rules"	This parameter contains the name of the Scale rule that the Cell is configured to use. The specific rule will therefore be used if a generator is configured to use the "Cell" Scale Rule. Special cases: to use the Mix rule, set the value to "? UseMixRule"; to use a random Mix

			rule, set the value to "Mix\n?"; to use a Random cell rule, set to "?"
	Harmony Rules	"Harmony Rules"	This parameter contains the name of the Harmony rule that the Cell is configured to use. The specific rule will therefore be used if a generator is configured to use the "Cell" Harmony Rule. Special cases: to use the Mix rule, set the value to "?UseMixRule"; to use a random Mix rule, set the value to "Mix\n?"; to use a Random cell rule, set to "?"
	Next Note Rules	"Next Note Rules"	This parameter contains the name of the Next Note rule that the Cell is configured to use. The specific rule will therefore be used if a generator is configured to use the "Cell" Next Note Rule. Special cases: to use the Mix rule, set the value to "?UseMixRule"; to use a random Mix rule, set the value to "Mix\n?"; to use a Random cell rule, set to "?"
	Rhythm Rules	"Rhythm Rules"	This parameter contains the name of the Rhythm rule that the Cell is configured to use. The specific rule will therefore be used if a generator/cell is configured to use the "Cell" Rhythm Rule. Special cases: to use the Mix rule, set the value to "?UseMixRule"; to use a random Mix rule, set the value to "Mix\n?"; to use a Random cell rule, set to "?"
<a href="#">Cell Meter</a>	Meter	"Meter"	"1:4", "2:4", "3:4", "4:4", "5:4", "6:4", "7:4", "8:4", "1:8", "2:8", "3:8", "4:8", "5:8", "6:8", "7:8", "8:8", "9:8", "10:8", "11:8", "12:8"

## Object: "Generator" (AKA Generator)

Parameter Group / View	Displayed Name	Script Parameter Name	Notes
<a href="#">Generator - Basics</a>	Name	"Generator"	<i>Generator name</i>
	Mute	"Mute"	Yes or No
	Patch	"Patch"	Value in range 0 to 127
	Use Patch?	"Use Patch?"	Yes or No

	MIDI Channel	"MIDI Channel"	Value in range 0, 16
	Generator Type	"Generator Type"	"Rhythmic", "Ambient", "Follower", "Repeater", "Patterns", "Listening"
	Pitch	"Pitch"	Value in range 0 to 127
	Pitch Range	"Pitch Range"	Value in range 11, 127
	Phrase Length	"Phrase Length"	Value in range 1, 256; also in <a href="#">Generator - Ambient</a>
	Phrase Length Range	"Phrase Length Range"	Value in range 0, 256; also in <a href="#">Generator - Ambient</a>
	Phrase Gaps	"Phrase Gaps"	Value in range 0, 256; also in <a href="#">Generator - Ambient</a>
	Phrase Gaps Range	"Phrase Gaps Range"	Value in range 0, 256; also in <a href="#">Generator - Ambient</a>
	Note Rest %	"Phrase Note Rest %"	Value in range 0, 100; also in <a href="#">Generator - Ambient</a>
<a href="#">Generator - Ambient</a>	Units	"Ambient Units"	"Seconds (thousandths of a)", "Beats (60ths of a)", "Full seconds"
	Duration	"Ambient Duration"	Value in range 0, 32000
	Duration Range	"Ambient Duration Range"	Value in range 0, 32000
	Gap Minimum	"Ambient Gap Min"	Value in range 0, 32000
	Gap Range	"Ambient Gap Range"	Value in range 0, 32000
<a href="#">Generator - Follower</a>	Follow Generator	"Follow Named Generator"	<i>Generator name</i>
	Percent	"Follow Percent"	Value in range 0, 100
	Strategy	"Follow Strategy"	"Chordal Harmony", "Interval Within Scale Rule", "Semitone Shift"
	Units	"Follow Delay Unit"	"Seconds (thousandths of a)", "Beats (60ths of a)", "Full seconds"
	Delay	"Follow Delay"	Value in range 0, 32000
	Delay Range	"Follow Delay Range"	Value in range 0, 32000
	Shift/Interval	"Follow Shift/Interval"	Value in range -60, +60
	S/I Range	"Follow Shift/Interval Range"	Value in range -60, +60
<a href="#">Generator - Repeater</a>	Generator	"Repeat Specific Generator"	<i>Generator name</i>
	Percent	"Repeat Bars Percent"	Value in range 0, 100
	Bars	"Repeat For Bars"	Value in range 1, 100
	History Range	"Repeat Bar History Range"	Value in range 0, 100
	History	"Repeat Bar History"	Value in range 1, 100
	Bars Range	"Repeat For Bars Range"	Value in range 0, 100
<a href="#">Generator - Patterns</a>	Patterns	"Patterns"	<i>Pattern String</i>
	Use Percent	"Patterns Use Percent"	Value in range 0, 100
	Mutation Factor	"Mutation factor"	Value from 0 to 1000 (corresponding to 0 to 100 percent)

	Bars Between	"Mutate No. Bars"	Value in range 0, 100
	Bars Range	"Mutate No. Bars Range"	Value in range 0, 100
	Mutate Rhythm?	"Mutation of Rhythm"	Yes or No
	Meter	"Meter"	"1:4", "2:4", "3:4", "4:4", "5:4", "6:4", "7:4", "8:4", "1:8", "2:8", "3:8", "4:8", "5:8", "6:8", "7:8", "8:8", "9:8", "10:8", "11:8", "12:8"
<b>Generator - Text-to-Music</b>	Text Pattern Enabled	"Text Pattern Enabled"	Yes or No
	Text Pattern Text	"Text Pattern Text"	<i>TTM Text String</i>
	Text Pattern Phrase Length	"Text Pattern Phrase Length"	Value in range 1 to 16
	Text Pattern Phrase Range	"Text Pattern Phrase Range"	Value in range 0 to 16
	Text Pattern Gaps	"Text Pattern Gaps"	Value in range 0 to 16
	Text Pattern Gaps Range	"Text Pattern Gaps Range"	Value in range 0 to 16
	Text Pattern Interval	"Text Pattern Interval"	Value in range 0 to 36
	Text Pattern Interval Range	"Text Pattern Interval Range"	Value in range 0 to 36
	Text Pattern Improvise After Tune	"Text Pattern Improvise After Tune"	Yes or No
	Text Pattern Variation	"Text Pattern Variation"	Value in range 0 to 100
	Text Pattern Repeats	"Text Pattern Repeats"	Value in range 0 to 16
	Text Pattern Repeats Range	"Text Pattern Repeats Range"	Value in range 0 to 16
	Text Pattern Tune Start At Index	"Text Pattern Tune Start At Index"	Value in range 0 to 100 No
	Text Pattern Tune Length Override	"Text Pattern Tune Length Override"	Value in range 0 to 100 No
	Text Pattern Display	"Text Pattern Display"	Yes or No
<b>Generator - Chording</b>	Depth	"Chord Depth"	Value in range 1, 32
	Pitch Offset	"Chord Pitch Offset"	Value in range -60, +60
	Delay	"Chord Delay"	Value in range 0, 32000
	Depth %	"Chord Depth Percent"	Value in range 0, 100
	Delay Unit	"Chord Delay Unit"	"Seconds (thousandths of a)", "Beats (60ths of a)", "Quantized Beats (60ths of a)"
	Velocity Factor	"Chord Velocity Factor"	Value in range -100, +100
	Delay Range	"Chord Delay Range"	Value in range 0, 32000
	Depth Range	"Chord Depth Range"	Value in range 0, 32
	Strategy	"Chord Strategy"	"Chordal Harmony", "Interval Within Scale Rule", "Semitone Shift"
	Shift/Interval	"Chord Shift/Interval"	Value in range -60, +60
	S/I Range	"Chord Shift/Interval Range"	Value in range -60, +60

Generator - Rules	Harmony Rules	"Harmony Rules"	Rule name. Special cases: to use the Mix rule, set the value to "? UseMixRule"; to use a random Mix rule, set the value to "Mix\n?"; to use a Random cell rule, set to "?"
	Next Note Rules	"Next Note Rules"	Rule name. Special cases: to use the Mix rule, set the value to "? UseMixRule"; to use a random Mix rule, set the value to "Mix\n?"; to use a Random cell rule, set to "?"
	Rhythm Rules	"Rhythm Rules"	Rule name. Special cases: to use the Mix rule, set the value to "? UseMixRule"; to use a random Mix rule, set the value to "Mix\n?"; to use a Random cell rule, set to "?"
	Scale Rules	"Scale Rules"	Rule name. Special cases: to use the Mix rule, set the value to "? UseMixRule"; to use a random Mix rule, set the value to "Mix\n?"; to use a Random cell rule, set to "?"
	Harmonize?	"Harmonize?"	Yes or No
	Voice Root	"Voice Root"	"A", "A#", "Ab" etc.
Generator - Comments	Copyright	"Copyright"	Copyright text
	Notes	"Notes"	Notes text
Generator - Articulation	Minimum	"Articulation Minimum"	Value in range 0, 100
	Range	"Articulation Range"	Value in range 0, 100
	Change	"Articulation Change"	Value in range 0, 100
	Change Range	"Articulation Change Range"	Value in range 0, 100
Generator - Controllers	Damper/Hold (64)	"Damper/Hold (64)"	Value in range -1, 127
	Harmonic Content (71)	"Harmonic Content (71)"	Value in range -1, 127
	Reverb (91)	"Reverb (91)"	Value in range -1, 127
	Chorus (93)	"Chorus (93)"	Value in range -1, 127
	Damper Release	"Damper Release"	Yes or No
	Portamento (84)	"Portamento (65)"	Value in range -1, 127
	MIDI Channel Sharing	"MIDI Channel Reallocation"	Yes or No
Generator - User Controller 1	MIDI CC	"User Controller 1 Midi Command"	Value in range 0 to 127
	Mode	"User Controller 1 Mode"	"-1 - Off", "0 - Random Drift", "1 - LFO (Min-Max-Min)", "2 - LFO (Max-Min-Max)", "3 - Sawtooth (Min-Max)", "4 - Sawtooth (Max-Min)"
	Minimum	"User Controller 1 Minimum"	Value in range 0, 127
	Range	"User Controller 1 Range"	Value in range 0, 127

	Change	"User Controller 1 Change"	Value in range 0, 127
	Change Range	"User Controller 1 Change Range"	Value in range 0, 127
	Update	"User Controller 1 Update"	Value in range 0, 10000
	Update Range	"User Controller 1 Update Range"	Value in range 0, 10000
	Update Units	"User Controller 1 Update Unit"	"Seconds (thousandths of a)", "Full seconds"
	Beat Cycle Length	"User Controller 1 Beat Cycle Length"	Value in range 0, 32000
	Phase Shift%	"User Controller 1 Phase Shift"	Value in range 0, 100
<b>Generator - User Controller 2</b>	MIDI CC	"User Controller 2 Midi Command"	Value in range 0 to 127
	Mode	"User Controller 2 Mode"	"-1 - Off", "0 - Random Drift", "1 - LFO (Min-Max-Min)", "2 - LFO (Max-Min-Max)", "3 - Sawtooth (Min-Max)", "4 - Sawtooth (Max-Min)"
	Minimum	"User Controller 2 Minimum"	Value in range 0, 127
	Range	"User Controller 2 Range"	Value in range 0, 127
	Change	"User Controller 2 Change"	Value in range 0, 127
	Change Range	"User Controller 2 Change Range"	Value in range 0, 127
	Update	"User Controller 2 Update"	Value in range 0, 10000
	Update Range	"User Controller 2 Update Range"	Value in range 0, 10000
	Beat Cycle Length	"User Controller 2 Beat Cycle Length"	Value in range 0, 32000
	Update Units	"User Controller 2 Update Unit"	"Seconds (thousandths of a)", "Full seconds"
	Phase Shift%	"User Controller 2 Phase Shift"	Value in range 0, 100
<b>Generator - Micro Note Delay</b>	Delay Range	"Micro Note Delay Range"	Value in range 0, 1000
	Delay Change	"Micro Note Delay Change"	Value in range 0, 1000
	Delay Offset	"Micro Note Delay Offset"	Value in range -1000, +1000
<b>Generator - Micro Pitch</b>	Bend Sensitivity	"Pitch Bend Sensitivity"	Value in range 0, 24
	Pitch Bend Offset	"Pitch Bend Offset"	Value in range -8192, +8191
	Pitch Range	"Micro Pitch Range"	Value in range 0, 8191
	Pitch Change	"Micro Pitch Change"	Value in range 0, 1000
	Pitch Update	"Micro Pitch Update"	Value in range 0, 10000
	Update Range	"Micro Pitch Update Range"	Value in range 0, 10000

Generator - Micro Vol Env	Range	"Micro Volume Range"	Value in range 0, 127
	Change	"Micro Volume Change"	Value in range 0, 127
	Update	"Micro Volume Update"	Value in range 0, 1000
	Update Range	"Micro Volume Update Range"	Value in range 0, 10000
Generator - Note to MIDI CC Mapping	CC for Note On?	"MIDI CC instead of Note?"	Yes or No
	Note On CC	"MIDI CC Note On Value"	Value from 0 to 127
	CC for Velocity?	"MIDI CC for Note On Velocity?"	Yes or No
	Velocity CC	"MIDI CC Note On Velocity"	Value from 0 to 127
	CC for Off?	"MIDI CC for Note Off?"	Yes or No
	Note Off CC	"MIDI CC Note Off Value"	Value from 0 to 127
Generator - User Envelope 1 (Volume)	MIDI CC	"User Envelope 1 MIDI CC"	Value from 0 to 127
	Enabled?	"User Envelope 1 Enabled"	Yes or No
	Envelope	"Volume"	<i>Envelope</i>
Generator - User Envelope 2 (Pan)	MIDI CC	"User Envelope 2 MIDI CC"	Value from 0 to 127
	Enabled?	"User Envelope 2 Enabled"	Yes or No
	Envelope	"Pan (10)"	<i>Envelope</i>
Generator - Envelope - Velocity	Velocity	"Velocity"	<i>Envelope</i>
Generator - Envelope - Velocity Range	Velocity Range	"Velocity Range"	<i>Envelope</i>
Generator - Envelope - Velocity Change	Velocity Change	"Velocity Change"	<i>Envelope</i>
Generator - Envelope - Velocity Change Range	Velocity Change Range	"Velocity Change Range"	<i>Envelope</i>

**<> Object: "Generator", "Scale Rule", "Harmony Rule", "Rhythm Rule", "Next Note Rule"**

Parameter Group / View	Displayed Name	Script Parameter Name	Notes
Scale Rule	Value	"Value"	A string like "1 0 0 0 0 0 0 0 0 0 0 0", where each item is a float value from 0 to 1, representing the rule weighting for each rule element from left to right



<a href="#">Harmony Rule</a>	Value	"Value"		A string like "1 0 0 0 1 0 0 0 0 0 0", where each item is a float value from 0 to 1, representing the rule weighting for each rule element from left to right
<a href="#">Rhythm Rule</a>	Value	"Value"		A string like "0 1 0 0 0 0 0 0 0", where each item is a float value from 0 to 1, representing the rule weighting for each rule element from left to right
<a href="#">Next Note Rule</a>	Value	"Value"		A string like "0 0 1 0 0 0 0 0 0 0 0 0", where each item is a float value from 0 to 1, representing the rule weighting for each rule element from left to right

## See Also

- [Scripting Overview](#)
- [Mix Script](#)
- [Cell Script](#)
- [Generator Script](#)
- [Wotja as a Hyperinstrument](#)

## Script Function Reference

### Utility Functions

- [wjsLog\(\)](#)
- [wjsGetVersion\(\)](#)
- [wjsGetRandom\(\)](#)
- [wjsRandomGetFromTo\(\)](#)

---

### Mix Functions

- `mixCellObjectCountGet()`
- `mixCellObjectIndexGet()`
- `mixCellObjectNameGet()`
- `mixCellObjectParameterGet()`
- `mixCellObjectParameterSet()`
- `mixCellSetTemplate()`
- `mixColumnLockGet()`
- `mixColumnLockSet()`
- `mixObjectCountGet()`
- `mixObjectIndexGet()`
- `mixObjectNameGet()`
- `mixObjectParameterGet()`
- `mixObjectParameterSet()`
- `mixParameterGet()`
- `mixParameterSet()`
- `mixTrackCellLoopGet()`
- `mixTrackCellLoopSet()`
- `mixTrackPanGet()`
- `mixTrackPanSet()`
- `mixTrackRuleGet()`
- `mixTrackRuleSet()`
- `mixTrackVolumeGet()`
- `mixTrackVolumeSet()`
- `mixVolumeGet()`
- `mixVolumeSet()`

## <> WME Functions

- `wmeParameterGet()`
- `wmeParameterSet()`
- `wmeGeneratorMIDIChannelGet()`
- `wmeRuleElementGet()`
- `wmeMIDISendCC()`
- `wmeBarDurationGet()`
- `wmeObjectCountGet()`
- `wmeObjectIndexGet()`
- `wmeObjectNameGet()`
- `wmeObjectParameterGet()`
- `wmeObjectParameterSet()`
- `wmeCellElapsedPercentGet()`
- `wmeRuleElementSet()`
- `wmeGeneratorEnvelopePercentGet()`
- `wmeGeneratorEnvelopePercentSet()`
- `wmeCellTrackIndexGet()`
- `wmeCellColumnIndexGet()`

## <> WAE Functions

- `waeGetFxNetworkIdForMix()`
- `waeGetFxNetworkIdForCellGlobal()`
- `waeGetFxNetworkIdForCellSynthTrackColumnChannel()`
- `waeGetFxNetworkIdForCellEffectTrackColumnChannel()`
- `waeFxNetworkAllPresetNamesGet()`
- `waeFxNetworkRandomPresetNamesGet()`
- `waeFxNetworkPresetGet()`
- `waeFxNetworkPresetSet()`

---

## <> WTE Functions

- [wteRandomWordsGet\(\)](#)
  - [wteShuffleWords\(\)](#)
  - [wteShuffleCharacters\(\)](#)
- 

## <> Utility functions

Wotja Script Utility functions all start with the *wjs* prefix.

### <> [wjsLog\(\)](#)

The function *wjsLog* may be used to display text in the [Script Console](#); this is very useful for debugging your scripts!

Example:

```
function onWmeStart() {  
    wjsLog("Hello World!")  
}
```

### <> [wjsGetRandom\(\)](#)

This function returns a random integer.

The function has no parameters.

Example:

```
var value = wjsGetRandom()
```

### <> [wjsGetVersion\(\)](#)

This returns the Wotja version as a string (e.g. "24.1.1") which can be useful for testing what features are available.

Example usage:

```
wjsLog("Wotja Version=" + wjsGetVersion())
```

## wjsRandomGetFromTo()

This function returns a random integer in the range of the two supplied values.

The function has two parameters:

- *minimum* the minimum value that can be returned
- *maximum* the maximum value that can be returned

Example:

```
function onWmeStart() {
  // Get a random integer between 0 and 50 inclusive.
  var value = wjsRandomGetFromTo(0, 50)
  wjsLog("random value=", value)
}
```

## Mix functions

These functions are intended to be used from [Mix Scripts](#). Wotja Script Mix functions all start with the `mix` prefix.

### mixCellObjectCountGet()

Returns the number of objects of the specified type, in the current Wotja Mix.

The function has the following parameters:

- *columnIndex* the column index of interest; from 0 to 3; 0 is the first cell on the left, 3 is the last cell on the right
- *trackIndex* the track index of interest; from 0 to 11; 0 is the first track at the top of the mix, 11 is the last track in the mix
- *objectType* the type of the object of interest, i.e. one of "Generator", "Scale Rule", "Harmony Rule", "Rhythm Rule", "Next Note Rule".

Example:

```
function testFunction() {
```

```

var columnIndex = 0
var trackIndex = 0
// Example usage:
var scaleRuleCount = mixCellObjectCountGet(columnIndex, trackIndex)
wjsLog("Mix cell(" + columnIndex + ", " + trackIndex + ")")
}

```

## `mixCellObjectIndexGet()`

Returns the index of the named object, which will be greater than or equal to 0 if the named object is found; the function will return -1 if not found.

The function has the following parameters:

- *columnIndex* the column index of interest; from 0 to 3; 0 is the first cell on the left, 3 is the last cell on the right
- *trackIndex* the track index of interest; from 0 to 11; 0 is the first track at the top of the mix, 11 is the last track in the mix
- *objectType* the type of the object of interest, i.e. one of "Generator", "Scale Rule", "Harmony Rule", "Rhythm Rule", "Next Note Rule".
- *objectName* the name of the object of interest

Example:

```

function testFunction() {
    var columnIndex = 0
    var trackIndex = 0
    // Example usage:
    var scaleRuleCount = mixCellObjectCountGet(columnIndex, trackIndex)
    wjsLog("Mix cell(" + columnIndex + ", " + trackIndex + ")")
    var scaleRuleName = mixCellObjectNameGet(columnIndex, trackIndex)
    wjsLog("Mix cell(" + columnIndex + ", " + trackIndex + ")")
    var indexOfScaleRuleName = mixCellObjectIndexGet(columnIndex, trackIndex, scaleRuleName)
    wjsLog("Mix cell(" + columnIndex + ", " + trackIndex + ")")
}

```

```
}
```

## `mixCellObjectNameGet()`

Returns the name of the specified object.

The function has the following parameters:

- *columnIndex* the column index of interest; from 0 to 3; 0 is the first cell on the left, 3 is the last cell on the right
- *trackIndex* the track index of interest; from 0 to 11; 0 is the first track at the top of the mix, 11 is the last track in the mix
- *objectType* the type of the object of interest, i.e. one of "Generator", "Scale Rule", "Harmony Rule", "Rhythm Rule", "Next Note Rule".
- *objectIndex* the index of the object of interest, starting at 0 for the first object.

Example:

```
function testFunction() {
  var columnIndex = 0
  var trackIndex = 0
  // Example usage:
  var scaleRuleCount = mixCellObjectCountGet(columnIndex, trackIndex)
  wjsLog("Mix cell(" + columnIndex + "," + trackIndex + ") has " + scaleRuleCount + " objects")
  var scaleRuleName = mixCellObjectNameGet(columnIndex, trackIndex, 0)
  wjsLog("Mix cell(" + columnIndex + "," + trackIndex + ") has object " + scaleRuleName)
}
```



## `mixCellObjectParameterGet()`

Returns the value of the parameter, for the specified object index of the specified object type, in the playing mix.

The function has the following parameters:

- *columnIndex* the column index of interest; from 0 to 3; 0 is the first cell on the left, 3 is the last cell on the right
- *trackIndex* the track index of interest; from 0 to 11; 0 is the first track at the top of the mix, 11 is the last track in the mix
- *objectType* the type of the object of interest, i.e. one of "Generator", "Scale Rule", "Harmony Rule", "Rhythm Rule", "Next Note Rule".
- *objectIndex* the index of the object of interest, starting at 0 for the first object. If the objectType is "Cell", or "Mix", you do not provide this parameter.
- *parameterName* the parameter of interest.

Example:

```
function testFunction() {
  // Example usage:
  var columnIndex = 0
  var trackIndex = 0
  var generatorIndex = 0
  var generatorValue = mixCellObjectParameterGet(columnIndex, trackIndex, "Generator")
  wjsLog("generatorValue = " + generatorValue)
  mixCellObjectParameterSet(columnIndex, trackIndex, "Generator", generatorValue)
  generatorValue = mixCellObjectParameterGet(columnIndex, trackIndex, "Generator")
  wjsLog("generatorValue = " + generatorValue)
  var cellValue = mixCellObjectParameterGet(columnIndex, trackIndex, "Cell")
  wjsLog("cellValue = " + cellValue)
}
```

### `mixCellObjectParameterSet()`

Sets the parameter to the the supplied value, for the specified object index of the specified object type, in the playing mix.

The function has the following parameters:

- *columnIndex* the column index of interest; from 0 to 3; 0 is the first cell on the left, 3 is the last cell on the right

- *trackIndex* the track index of interest; from 0 to 11; 0 is the first track at the top of the mix, 11 is the last track in the mix
- *objectType* the type of the object of interest, i.e. one of "Generator", "Scale Rule", "Harmony Rule", "Rhythm Rule", "Next Note Rule".
- *objectIndex* the index of the object of interest, starting at 0 for the first object. If the objectType is "Cell", or "Mix", you do not provide this parameter.
- *parameterName* the parameter of interest.
  - If for parameterName you use "Rhythm Rules", "Scale Rules", "Harmony Rules", "Next Note Rules" [Note the PLURAL] then the method will set the Generator (or Cell) to the \*Cell\* rule with name specified in newValue.
  - There is a special case for this method, which is used \*only\* if objectType is "Generator" and it works as follows:

If for parameterName you use one of the following special values "Scale Rule", "Rhythm Rule", "Harmony Rule", "Next Note Rule" [note the SINGULAR] then the method will look for a cell rule matching the generator name (if not found, it will create that cell rule) and attach the generator (if not already attached) to that cell rule rule value as specified.
- *newValue* the new parameter value to use.
  - If for parameterName you use one of the following values for "Rhythm Rules", "Scale Rules", "Harmony Rules", "Next Note Rules" [Note the PLURAL]
    - To attach to a \*Cell\* rule, then simply use the Rule Name for newValue. e.g. "My Cell Rule Name"
    - To attach to a \*Mix\* rule, then prefix the rule name (for newValue) with "Mix\n". e.g. "Mix\nMy Mix Rule Name"
    - Other special cases: to use the Mix rule, set the value to "?UseMixRule"; to use a random Mix rule, set the value to "Mix\n?"; to use a Random cell rule, set to "?"

Example:

```
function testFunction() {
  // Example usage:
  var columnIndex = 0
  var trackIndex = 0
  var generatorIndex = 0
  var generatorValue = mixCellObjectParameterGet(columnIndex, trackIndex, generatorIndex)
  wjsLog("generatorValue = " + generatorValue)
  mixCellObjectParameterSet(columnIndex, trackIndex, "Generator", generatorValue)
```



```

generatorValue = mixCellObjectParameterGet(columnIndex
wjsLog("generatorValue = " + generatorValue)
var cellValue = mixCellObjectParameterGet(columnIndex,
wjsLog("cellValue = " + cellValue)
mixCellObjectParameterSet(columnIndex, trackIndex, "Ce
cellValue = mixCellObjectParameterGet(columnIndex, trac
wjsLog("cellValue = " + cellValue)

// Showing use of the special case:
mixCellObjectParameterGet(columnIndex, trackIndex, "Ger
mixCellObjectParameterGet(columnIndex, trackIndex, "Ger
mixCellObjectParameterGet(columnIndex, trackIndex, "Ger
mixCellObjectParameterGet(columnIndex, trackIndex, "Ger
}

```

## `mixCellSetTemplate()`

Use this to set the Template to load. You can either use a built-in Template or a custom [Template URI](#) (see the forgoing link for details on how to export them to clipboard).

The function has the following parameters:

- *columnIndex* the column index of interest; from 0 to 3; 0 is the first cell on the left, 3 is the last cell on the right
- *trackIndex* the track index of interest; from 0 to 11; 0 is the first track at the top of the mix, 11 is the last track in the mix
- *templateData* the template to use. You can provide either a path to a built-in template, or the Wotja URI for a custom item/li>

Example usage with Paths:

```

// Use a built-in Template
mixCellTemplateSet(0, 0, "intermorphic/wotja/paks/IM_ZG_I
mixCellTemplateSet(0, 0, "intermorphic/mixtikl/paks/BP22

```

```
// Shorter alternative to the above (easier to read):
mixCellTemplateSet(0, 0, "im/IM_ZG_MorphingDrum+Bass/AMB:
mixCellTemplateSet(0, 0, "im/BP22 Drums/Clave and toms.nc

// Use a custom Template:
mixCellTemplateSet(0, 0, "wotja://template/DATA...")
mixCellTemplateSet(0, 0, "noatikl://DATA...")
```

### `mixColumnLockGet()`

Returns the currently locked column index: if no column is currently locked, this returns -1 otherwise, returns a value from 0 to 3

The function has no parameters

Example:

```
function testFunction() {
  var columnLockedIndex = mixColumnLockGet()
  wjsLog("columnLockedIndex = " + columnLockedIndex)
}
```

### `mixColumnLockSet()`

Sets column lock (or not); if columnIndex is 0 to 3, this sets the column lock on; if columnIndex is -1, this turns the column lock off

The function has the following parameters:

- *columnIndex* The columnIndex index of the cell (from 0 to 3).

Example:

```
function testFunction() {
  // Example usages:
```

```

var columnLockedIndex = mixColumnLockGet()
wjsLog("columnLockedIndex = " + columnLockedIndex)

mixColumnLockSet(0)
columnLockedIndex = mixColumnLockGet()
wjsLog("columnLockedIndex = " + columnLockedIndex)

mixColumnLockSet(1)
columnLockedIndex = mixColumnLockGet()
wjsLog("columnLockedIndex = " + columnLockedIndex)

mixColumnLockSet(2)
columnLockedIndex = mixColumnLockGet()
wjsLog("columnLockedIndex = " + columnLockedIndex)

mixColumnLockSet(3)
columnLockedIndex = mixColumnLockGet()
wjsLog("columnLockedIndex = " + columnLockedIndex)

mixColumnLockSet(-1)
columnLockedIndex = mixColumnLockGet()
wjsLog("columnLockedIndex = " + columnLockedIndex)
}

```

## `mixObjectCountGet()`

Returns the number of objects of the specified type, in the current Wotja Mix.

The function has the following parameters:

- *objectType* the type of the object of interest, i.e. one of "Generator", "Scale Rule", "Harmony Rule", "Rhythm Rule", "Next Note Rule".

Example:

```
function testFunction() {
  // Example usage:
  var scaleRuleCount = mixObjectCountGet("Scale Rule")
  wjsLog("Mix scale rule count = " + scaleRuleCount)
}
```

### `mixObjectIndexGet()`

Returns the index of the named object, which will be greater than or equal to 0 if the named object is found; the function will return -1 if not found.

The function has the following parameters:

- *objectType* the type of the object of interest, i.e. one of "Generator", "Scale Rule", "Harmony Rule", "Rhythm Rule", "Next Note Rule".
- *objectName* the name of the object of interest

Example:

```
function testFunction() {
  // Example usage:
  var scaleRuleName = mixObjectNameGet("Scale Rule", 0)
  wjsLog("Mix scale rule 0 has name = " + scaleRuleName)
  var indexOfScaleRuleName = mixObjectIndexGet("Scale Rule", scaleRuleName)
  wjsLog("Mix scale rule with name=" + scaleRuleName + " has index=" + indexOfScaleRuleName)
}
```

### `mixObjectNameGet()`

Returns the name of the specified object.

The function has the following parameters:

- *objectType* the type of the object of interest, i.e. one of "Generator", "Scale Rule", "Harmony Rule", "Rhythm Rule", "Next Note Rule".

- *objectIndex* the index of the object of interest, starting at 0 for the first object.

Example:

```
function testFunction() {
  // Example usage:
  var scaleRuleCount = mixObjectCountGet("Scale Rule")
  wjsLog("Mix scale rule count = " + scaleRuleCount)
  var scaleRuleName = mixObjectNameGet("Scale Rule", 0)
  wjsLog("Mix scale rule 0 has name = " + scaleRuleName)
  var indexOfScaleRuleName = mixObjectIndexGet("Scale Rule", 0)
  wjsLog("Mix scale rule with name=" + scaleRuleName + "
}
}
```

## `mixObjectParameterGet()`

Returns the value of the parameter, for the specified object index of the specified object type, in the playing mix.

The function has the following parameters:

- *objectType* the type of the object of interest, i.e. one of "Generator", "Scale Rule", "Harmony Rule", "Rhythm Rule", "Next Note Rule".
- *objectIndex* the index of the object of interest, starting at 0 for the first object. If the *objectType* is "Cell", or "Mix", you do not provide this parameter.
- *parameterName* the parameter of interest.

Example:

```
function testFunction() {
  // Example usage:
  var scaleRuleValue = mixObjectParameterGet("Scale Rule", 0)
  wjsLog("Mix scale rule index 0, value was = " + scaleRuleValue)
  scaleRuleValue = mixObjectParameterSet("Scale Rule", 0, "C")
  scaleRuleValue = mixObjectParameterGet("Scale Rule", 0)
}
```

```
wjsLog("Mix scale rule index 0, value now = " + scaleRu
}
```

## `mixObjectParameterSet()`

Sets the parameter to the the supplied value, for the specified object index of the specified object type, in the playing mix.

The function has the following parameters:

- *objectType* the type of the object of interest, i.e. one of "Generator", "Scale Rule", "Harmony Rule", "Rhythm Rule", "Next Note Rule".
- *objectIndex* the index of the object of interest, starting at 0 for the first object. If the *objectType* is "Cell", or "Mix", you do not provide this parameter.
- *parameterName* the parameter of interest.
- *newValue* the new parameter value to use.  
Special behavior for Rules: if you set *newValue* to be the name a built-in rule (e.g. "Major") for the parameter value, the parameter value will be set magically to that of the corresponding rule type. Otherwise, you'd need to set the rule value in the normal way, with a string looking something like "1 0 0.5 0 1 0.5 0 1 0 0.5 0 0.5"

Example:

```
function testFunction() {
  // Example usage:
  var scaleRuleValue = mixObjectParameterGet("Scale Rule'
  wjsLog("Mix scale rule index 0, value was = " + scaleRu
  scaleRuleValue = mixObjectParameterSet("Scale Rule", 0
  scaleRuleValue = mixObjectParameterGet("Scale Rule", 0
  wjsLog("Mix scale rule index 0, value now = " + scaleRu
}
```

## `mixParameterGet()`

This function returns the value of the named parameter, for the object within whose trigger script you place this call. Note that the value returned is always a *string*, which you can convert to a number using *tonumber()*.

The function has one parameter:

- *parameterName* the name of the parameter for which you want to determine the current value.

Example:

```
function testFunction() {
  // Example usage:
  var mixValue = mixParameterGet("Scale Rules")
  wjsLog("Mix: Scale Rules = " + mixValue)

  mixValue = mixParameterGet("Mix Root")
  wjsLog("Mix: Mix Root = " + mixValue)

  mixValue = mixParameterGet("Tempo")
  wjsLog("Mix: Tempo = " + mixValue)
}
```

## `mixParameterSet()`

This function is used to set the mix's parameter to the supplied value, for the object within whose trigger script you place this call.

The function has two parameters:

- *parameterName* the name of the parameter for which you want to set the value.
- *newValue* the new parameter value to use.

Special behavior for Rules: if you set *newValue* to be the name a built-in rule for the parameter value (e.g. [Scale Rule "All Scale Major"](#)), the parameter value will be set magically to that of the corresponding rule type. Otherwise, you'd need to set the rule value in the normal way, with a string looking something like `"1 0 1 0 1 1 0 1 0 1 0 1"`

Example:

```
function testFunction() {
  // Example usage:
  var mixValue = mixParameterGet("Scale Rules")
  wjsLog("Mix: Scale Rules = " + mixValue)
  mixParameterSet("Scale Rules", "Default")
  mixValue = mixParameterGet("Scale Rules")
  wjsLog("Mix: Scale Rules = " + mixValue)

  mixValue = mixParameterGet("Mix Root")
  wjsLog("Mix: Mix Root = " + mixValue)
  mixParameterSet("Mix Root", "C")
  mixValue = mixParameterGet("Mix Root")
  wjsLog("Mix: Mix Root = " + mixValue)

  mixValue = mixParameterGet("Tempo")
  wjsLog("Mix: Tempo = " + mixValue)
  mixParameterSet("Tempo", "100")
  mixValue = mixParameterGet("Tempo")
  wjsLog("Mix: Tempo = " + mixValue)
}
```

### `mixTrackCellLoopGet()`

Returns a value which is the column currently locked for that track; from 0 to 3; or -1 if no cell is looping

The function has the following parameters:

- *trackIndex* the track index of interest; from 0 to 11; 0 is the first track at the top of the mix, 11 is the last track in the mix

Example:



```
function testFunction() {
  var trackIndex = 0
  var loopingCellColumnIndex = mixTrackCellLoopGet(trackIndex)
  wjsLog("loopingCellColumnIndex = " + loopingCellColumnIndex)
}
```

## `mixTrackCellLoopSet()`

sets cell to loop (or not); if columnIndex is 0 to 3, this starts the track looping at that column index; if columnIndex is -1, this stops the track looping; what happens next, depends on the track rule.

The function has the following parameters:

- *trackIndex* the track index of interest; from 0 to 11; 0 is the first track at the top of the mix, 11 is the last track in the mix
- *columnIndex* The columnIndex index of the cell (from 0 to 3).

Example:

```
function testFunction() {
  var trackIndex = 0
  var loopingCellColumnIndex = mixTrackCellLoopGet(trackIndex)
  wjsLog("loopingCellColumnIndex = " + loopingCellColumnIndex)

  mixTrackCellLoopSet(trackIndex, -1)
  loopingCellColumnIndex = mixTrackCellLoopGet(trackIndex)
  wjsLog("loopingCellColumnIndex = " + loopingCellColumnIndex)
}
```

## `mixTrackPanGet()`

Returns pan value from 0 to 127; 0 is far left, 64 is middle, 127 is far right

The function has the following parameters:

- *trackIndex* the track index of interest; from 0 to 11; 0 is the first track at the top of the mix, 11 is the last track in the mix

Example:

```
function testFunction() {
  // Example usage:
  var trackIndex = 0
  var value = mixTrackPanGet(trackIndex) // Value from 0
  wjsLog("Track pan = " + value)
}
```



## `mixTrackPanSet()`

Sets the pan setting for the specified track.

The function has the following parameters:

- *trackIndex* the track index of interest; from 0 to 11; 0 is the first track at the top of the mix, 11 is the last track in the mix
- *panValue* The pan value to use, from 0 to 127; 0 is far left, 64 is middle, 127 is far right

Example:

```
function testFunction() {
  // Example usage:
  var trackIndex = 0
  var value = mixTrackPanGet(trackIndex) // Value from 0
  wjsLog("Track pan = " + value)
  mixTrackPanSet(trackIndex, 64) // Set to middle!
  value = mixTrackPanGet(trackIndex)
  wjsLog("Track pan = " + value)
}
```

```
}
```

## `mixTrackRuleGet()`

Returns a string value which is current track rule setting, one of: "Sequence", "Loop" or "One Shot"

The function has the following parameters:

- *trackIndex* the track index of interest; from 0 to 11; 0 is the first track at the top of the mix, 11 is the last track in the mix

Example:

```
function testFunction() {  
  
    var trackIndex = 0  
  
    var trackRule = mixTrackRuleGet(trackIndex)  
    wjsLog("trackRule = " + trackRule)  
}
```

## `mixTrackRuleSet()`

Sets the pan setting for the specified track.

The function has the following parameters:

- *trackIndex* the track index of interest; from 0 to 11; 0 is the first track at the top of the mix, 11 is the last track in the mix
- *ruleValue* one of: "Sequence", "Loop" or "One Shot"

Example:

```
function testFunction() {
```

```

var trackIndex = 0

var trackRule = mixTrackRuleGet(trackIndex)
wjsLog("trackRule = " + trackRule)

mixTrackRuleSet(trackIndex, "One Shot")
trackRule = mixTrackRuleGet(trackIndex)
wjsLog("trackRule = " + trackRule)

mixTrackRuleSet(trackIndex, "Loop")
trackRule = mixTrackRuleGet(trackIndex)
wjsLog("trackRule = " + trackRule)

mixTrackRuleSet(trackIndex, "Sequence")
trackRule = mixTrackRuleGet(trackIndex)
wjsLog("trackRule = " + trackRule)
}

```

## `mixTrackVolumeGet()`

Returns volume value from 0 to 127

The function has the following parameters:

- *trackIndex* the track index of interest; from 0 to 11; 0 is the first track at the top of the mix, 11 is the last track in the mix

Example:

```

function testFunction() {
  // Example usage:
  var trackIndex = 0
  var value = mixTrackVolumeGet(trackIndex)
  wjsLog("Track volume = " + value)
}

```

```
}
```

## `mixTrackVolumSet()`

Sets the volume setting for the specified track.

The function has the following parameters:

- *trackIndex* the track index of interest; from 0 to 11; 0 is the first track at the top of the mix, 11 is the last track in the mix
- *volumeValue* The volume value to use, from 0 to 127

Example:

```
function testFunction() {  
  // Example usage:  
  var trackIndex = 0  
  var value = mixTrackVolumeGet(trackIndex)  
  wjsLog("Track volume = " + value)  
  mixTrackVolumeSet(trackIndex, 127) // Set to max!  
  value = mixTrackVolumeGet(trackIndex)  
  wjsLog("Track volume = " + value)  
}
```

## `mixVolumeGet()`

Returns the mix master volume value from 0 to 127

The function has no parameters

Example:

```
function testFunction() {  
  // Example usage:  
  var trackIndex = 0  
  var value = mixVolumeGet(trackIndex)
```

```
wjsLog("mix volume = " + value)
}
```

## `mixVolumeSet()`

Sets the the mix master volume setting

The function has the following parameters:

- *volumeValue* The volume value to use, from 0 to 127

Example:

```
function testFunction() {
  // Example usage:
  var trackIndex = 0
  var value = mixVolumeGet(trackIndex)
  wjsLog(" volume = " + value)
  mixVolumeSet(trackIndex, 127) // Set to max!
  value = mixVolumeGet(trackIndex)
  wjsLog("mix volume = " + value)
}
```

## **WME functions**

The WME contains many powerful scripting features. The WME related functions all start with the `wme` prefix.

### `wmeParameterGet()`

This function returns the value of the named parameter, for the object within whose trigger script you place this call. Note that the value returned is always a *string*, which you can convert to a number using `tonumber()`.

The function has one parameter:

- *parameterName* the name of the parameter for which you want to determine the current value.

Example:

```
function onWmeStart() {
  // Get the current Chord Depth (assuming we make this call
  // Generator's trigger script!)
  var chordDepth = wmeParameterGet("Chord Depth")
  wjsLog("Chord Depth=" + chordDepth)
}
```



### `wmeParameterSet()`

This function is used to set the object's parameter to the supplied value, for the object within whose trigger script you place this call.

The function has two parameters:

- *parameterName* the name of the parameter for which you want to set the value.
- *newValue* the new parameter value to use.  
Special behavior for Rules: if you set *newValue* to be the name a built-in rule (e.g. "Major") for the parameter value, the parameter value will be set magically to that of the corresponding rule type. Otherwise, you'd need to set the rule value in the normal way, with a string looking something like "1 0 0.5 0 1 0.5 0 1 0 0.5 0 0.5"

Example:

```
function onWmeStart() {
  // Set the Chord Depth to 4 (assuming we make this call
  // Generator's trigger script!)
  var chordDepth = 4
  wmeParameterSet("Chord Depth", chordDepth)
}
```

```
}
```

## `wmeGeneratorMIDIChannelGet()`

This function returns the MIDI channel (assuming the script is called from within a generator script!). The value returned is a value from 0 to 15 (MIDI channel is actually displayed in the WME Network screen as 1 to 16).

The function has no parameters.

Example:

```
function onWmeStart() {
    var channel = wmeGeneratorMIDIChannelGet ()
    wjsLog("Channel=" + channel)
}
```

## `wmeRuleElementGet()`

This function returns the rule element value of the specified index, for the indexed rule object. The value returned is a value between 0 and 127.

The function has the following parameters:

- *objectType* the type of the object of interest, i.e. one of "Generator", "Scale Rule", "Harmony Rule", "Rhythm Rule", "Next Note Rule".
- *ruleObjectIndex* the index of the rule object of interest, starting at 0 for the first object.
- *ruleElementIndex* the index of the element; starting at 0 for the initial element.

Example 1:

```
function onWmeStart() {
    // Dump out the scale rule element values, for the first
    var scaleRuleIndex = 0
    var itemIndex = 0
    while (itemIndex < 12)
```



```

{
    var value = wmeRuleElementGet ("Scale Rule", scaleRu
    wjsLog("value=" + value)
    index = itemIndex + 1
}
}

```

Example 2:

```

function onWmeStart() {
    var scaleRuleCount = wmeObjectCountGet("Scale Rule")
    wjsLog("Scale Rule objects=" + scaleRuleCount)
    for (var scaleRuleIndex = 0; scaleRuleIndex < scaleRule
        // Dump out the scale rule element values, for the s
        var ruleName = wmeObjectNameGet("Scale Rule", scaleRu
        wjsLog("Scale=" + ruleName)

        var itemIndex = 0
        while (itemIndex < 12)
        {
            var value = wmeRuleElementGet ("Scale Rule", scalef
            wjsLog("itemIndex" + ", " + itemIndex + ", " + "\
            itemIndex = itemIndex + 1
        }
    }
}

```

 [wmeMIDISendCC\(\)](#)

This function emits the specified MIDI CC event.

The function has the following parameters:

- *channel* the MIDI channel to use; from 1 to 16.
- *cc* the MIDI CC to use; from 0 to 127.
- *value* the controller value to use, from 0 to 127.
- *delay* an optional delay to use, from 0 up, defaulting to 0; in WME pattern time units (where 60 represents one crotchet or quarter note). The delay is relative to the current WME timebase relevant to the trigger script in question.

Example:

```
function onWmeBar(bar) {
    // Generator trigger...
    // Apply a pan sweep through the bar from left to right
    // to show-off the use of wmeMIDISendCC.

    var duration = wmeBarDurationGet()

    var midiChannel = wmeGeneratorMIDIChannelGet()
    var delay = 0
    var ccController = 10 // Pan controller!
    var value = 0
    while delay <= duration {
        var value = (127 * delay) / duration
        // Note that the "delay" is optional; we use this in
        // demo, to get a sweep effect from start to end of t
        wmeMIDISendCC(midiChannel, ccController, value, delay)
        delay = delay + 20
    }
}
```



 `wmeBarDurationGet ()`

This function returns the duration of the mix bar, in WME pattern time units; where 240 represents four crotchets or one whole note or a bar of 4:4 time.

The function has no parameters.

Example:

```
function onWmeStart() {
  var duration = wmeBarDurationGet()
  wjsLog("duration=" + duration)
}
```

### `wmeObjectCountGet()`

Returns the number of objects of the specified type, in the current Wotja Mix.

The function has the following parameters:

- *objectType* the type of the object of interest, i.e. one of *"Generator"*, *"Scale Rule"*, *"Harmony Rule"*, *"Rhythm Rule"*, *"Next Note Rule"*

Example:

```
function onWmeStart() {
  var generatorCount = wmeObjectCountGet("Generator")
  wjsLog("Generator objects=" + generatorCount)
}
```

### `wmeObjectIndexGet()`

Returns the index of the named object, which will be greater than or equal to 0 if the named object is found; the function will return -1 if not found.

The function has the following parameters:

- *objectType* the type of the object of interest, i.e. one of "Generator", "Scale Rule", "Harmony Rule", "Rhythm Rule", "Next Note Rule".
- *objectName* the name of the object of interest

Example:

```
function onWmeStart() {
    var majorScaleRuleIndex = wmeObjectIndexGet("Scale Rule")
    wjsLog("majorScaleRuleIndex=" + majorScaleRuleIndex)
}
```



### wmeObjectNameGet()

Returns the name of the specified object.

The function has the following parameters:

- *objectType* the type of the object of interest, i.e. one of "Generator", "Scale Rule", "Harmony Rule", "Rhythm Rule", "Next Note Rule".
- *objectIndex* the index of the object of interest, starting at 0 for the first object.

Example:

```
function onWmeStart() {
    var generatorCount = wmeObjectCountGet("Generator")
    wjsLog("Generator objects=" + generatorCount)

    for (var index = 0; index < generatorCount; index++) {
        wjsLog("Generator name=" + wmeObjectNameGet("Generator", index))
    }
}
```

```
}
```

## `wmeObjectParameterGet()`

Returns the value of the parameter, for the specified object index of the specified object type, in the playing mix.

The function has the following parameters:

- *objectType* the type object of interest, i.e. one of "Mix", "Cell", "Generator", "Scale Rule", "Harmony Rule", "Rhythm Rule", "Next Note Rule".
- *objectIndex* the index of the object of interest, starting at 0 for the first object. If the *objectType* is "Cell", or "Mix", you do not provide this parameter.
- *parameterName* the parameter of interest.

Example:

```
function onWmeStart() {  
  
    var valueWas = wmeObjectParameterGet("Generator", 0, '  
    wmeObjectParameterSet("Generator", 0, "Chord Depth", 4);  
  
    var valueNow = wmeObjectParameterGet("Generator", 0, '  
    wjsLog("Chord Depth valueWas=" + valueWas + ", valueNow  
}
```

## `wmeObjectParameterSet()`

Sets the parameter to the the supplied value, for the specified object index of the specified object type, in the playing mix.

The function has the following parameters:

- *objectType* the type object of interest, i.e. one of "Mix", "Cell", "Generator", "Scale Rule", "Harmony Rule", "Rhythm Rule", "Next Note Rule"
- *objectIndex* the index of the object of interest, starting at 0 for the first object. If the *objectType* is "Cell", or "Mix", you do not provide this parameter.
- *parameterName* the parameter of interest.
  - If for *parameterName* you use "Rhythm Rules", "Scale Rules", "Harmony Rules", "Next Note Rules" [Note the PLURAL] then the method will set the Generator (or Cell) to the \*Cell\* rule with name specified in *newValue*.
  - There is a special case for this method, which is used \*only\* if *objectType* is "Generator" and it works as follows:  
If for *parameterName* you use one of the following special values "Scale Rule", "Rhythm Rule", "Harmony Rule", "Next Note Rule" [note the SINGULAR] then the method will look for a cell rule matching the generator name (if not found, it will create that cell rule) and attach the generator (if not already attached) to that cell rule rule value as specified.
- *newValue* the new parameter value to use.
  - If for *parameterName* you use one of the following values for "Rhythm Rules", "Scale Rules", "Harmony Rules", "Next Note Rules" [Note the PLURAL]
    - To attach to a \*Cell\* rule, then simply use the Rule Name for *newValue*. e.g. "My Cell Rule Name"
    - To attach to a \*Mix\* rule, then prefix the rule name (for *newValue*) with "Mix\n". e.g. "Mix\nMy Mix Rule Name"
    - Other special cases: to use the Mix rule, set the value to "?UseMixRule"; to use a random Mix rule, set the value to "Mix\n?"; to use a Random cell rule, set to "?"

Example:

```
function onWmeStart() {

    var valueWas = wmeObjectParameterGet("Generator", 0, '
wmeObjectParameterSet("Generator", 0, "Chord Depth", 4);

    var valueNow = wmeObjectParameterGet("Generator", 0, '
    wjsLog("Chord Depth valueWas=" + valueWas + ", valueNow
```

```
// Showing use of the special case:
var generatorIndex = 0
wmeObjectParameterSet("Generator", generatorIndex, "Scale")
wmeObjectParameterSet("Generator", generatorIndex, "Rhythm")
wmeObjectParameterSet("Generator", generatorIndex, "Next Note")
wmeObjectParameterSet("Generator", generatorIndex, "Harmony")
}
```

### wmeCellElapsedPercentGet ()

Returns the elapsed percentage of the playing mix, within the context of it's container cell. A cell with duration set to "Infinite" is actually around 8 hours before it restarts; so will tend to report 0 percent. A cell with duration of just a few bars, will progress quickly from 0 through to 100.

The function has no parameters.

Example:

```
function onWmeBar() {
  var value = wmeCellElapsedPercentGet()
  wjsLog("wmeCellElapsedPercentGet=" + value)
}
```

### wmeRuleElementSet ()

This function set the rule element value of the specified index, for the indexed rule object, to the specified value. The value must be between 0 and 127.

The function has the following parameters:

- *objectType* the type of the object of interest, i.e. one of "Generator", "Scale Rule", "Harmony Rule", "Rhythm Rule", "Next Note Rule".
- *ruleObjectIndex* the index of the rule object of interest, starting at 0 for the first object.
- *ruleElementIndex* the index of the element; starting at zero for the initial element.

- *newValue* the new element value to use, in a range from 0 to 127.

Example 1:

```
function onWmeStart() {
  // Set the scale rule element values, to odd settings
  var scaleRuleIndex = 0
  var itemIndex = 0
  while (itemIndex < 12) {
    wmeRuleElementSet ("Scale Rule", scaleRuleIndex, iter
    itemIndex = itemIndex + 1
  }
}
```



Example 2:

```
function onWmeStart() {
  var ruleCount = wmeObjectCountGet("Scale Rule")
  wjsLog("Scale Rule objects=" + ruleCount)

  for (var scaleRuleIndex = 0; scaleRuleIndex < ruleCount; scaleRuleIndex++) {
    // Set the scale rule element values, for the scale,
    var ruleName = wmeObjectNameGet("Scale Rule", scaleRuleIndex)
    wjsLog("Scale=" + ruleName)

    var itemIndex = 0
    while (itemIndex < 12) {
      var value = (itemIndex * 127) / 12
      wmeRuleElementSet ("Scale Rule", scaleRuleIndex, itemIndex, value)
      itemIndex = itemIndex + 1
    }
  }
}
```



```
}
```

## `wmeGeneratorEnvelopePercentGet()`

This function returns the percent value for the specified envelope at the given percent position. The value returned is a value between 0 and 127.

The function has the following parameters:

- *generatorObjectIndex* the index of the generator object of interest.
- *parameterName* the name of the envelope parameter (e.g. *Volume*).
- *percent* the percent value; from 0 to 100.

Example:

```
function onWmeStart() {
    var generatorCount = wmeObjectCountGet("Generator")
    wjsLog("Generator objects=" + generatorCount)

    for (var generatorIndex = 0; generatorIndex < generatorCount; generatorIndex++)
        // Ramp-up every generator volume envelope!
        var generatorName = wmeObjectNameGet("Generator", generatorIndex)
        wjsLog("Generator=" + generatorName)

        var percent = 0
        while (percent <= 100)
        {
            var valueWas = wmeGeneratorEnvelopePercentGet (generatorIndex, percent)
            wjsLog("Volume at " + percent + ", " + generatorName + " was=" + valueWas)

            var setToValue = percent + 1
            wjsLog("set to new value =" + setToValue)
            wmeGeneratorEnvelopePercentSet (generatorIndex, "Volume", setToValue)
        }
    }
}
```

```

    var valueNow = wmeGeneratorEnvelopePercentGet (generatorIndex, parameterName, percent)
    wjsLog("Volume at " + percent + ", " + "now=" + valueNow)
    percent = percent + 1
  }
}
}

```

### `wmeGeneratorEnvelopePercentSet()`

This function sets the envelope value at the specified percent, for the indexed generator, to the specified value. The value must be between 0 and 127.

The function has the following parameters:

- *generatorObjectIndex* the index of the generator object of interest.
- *parameterName* the name of the envelope parameter (e.g. *Volume*).
- *percent* the percent value; from 0 to 100.
- *newValue* the new element value to use, in a range from 0 to 127.

Example:

```

function onWmeStart() {
  var generatorCount = wmeObjectCountGet("Generator")
  wjsLog("Generator objects=" + generatorCount)

  for (var generatorIndex = 0; generatorIndex < generatorCount; generatorIndex++)
    // Ramp-up every generator volume envelope!
    var generatorName = wmeObjectNameGet("Generator", generatorIndex)
    wjsLog("Generator=" + generatorName)

  var percent = 0
  while (percent <= 100)
  {

```

```

var valueWas = wmeGeneratorEnvelopePercentGet (generatorIndex, percent)
wjsLog("Volume at " + percent + ", " + "was=" + valueWas)

var setToValue = percent + 1
wjsLog("set to new value =" + setToValue)
wmeGeneratorEnvelopePercentSet (generatorIndex, "Volume", setToValue)

var valueNow = wmeGeneratorEnvelopePercentGet (generatorIndex, percent)
wjsLog("Volume at " + percent + ", " + "now=" + valueNow)

percent = percent + 1
}
}
}

```

#### **wmeCellTrackIndexGet()**

This function returns the track index of the Mix Cell containing the current generator/mix. This is used with various WAE related functions.

The function has no parameters

Example:

```

function onWmeStart() {

    var cellTrackIndex = wmeCellTrackIndexGet()

}

```

#### **wmeCellColumnIndexGet()**

This function returns the column index of the Mix Cell containing the current generator/mix. This is used with various WAE related functions.

The function has no parameters

Example:

```
function onWmeStart() {  
  
    var cellColumnIndex = wmeCellColumnIndexGet()  
  
}
```

## WAE functions

WAE related functions all start with the `wae` prefix.

### `waeGetFxNetworkIdForMix()`

This function returns the id that can be used for manipulating the effects chain for the Global Mix FX.

The function has no parameters

Example:

```
function onWmeStart() {  
  
    var fxNetworkIdForMix = waeGetFxNetworkIdForMix()  
  
}
```

### `waeGetFxNetworkIdForCellGlobal()`

This function returns the id that can be used for manipulating the effects chain for the Cell, identified by the supplied track and column indexes.

The function has the following parameters:

- `trackIndex` The track index of the cell (from 0 to 11).

- *columnIndex* The columnIndex index of the cell (from 0 to 3).

Example:

```
function onWmeStart() {  
  
    var fxNetworkIdForCellAtTrack0Column1 = waeGetFxNetworkIdForCellAtTrack0Column1()  
  
}
```

### `waeGetFxNetworkIdForCellSynthTrackColumnChannel()`

This function returns the id that can be used for manipulating the effects chain for the Cell, identified by the supplied track and column indexes.

The function has the following parameters:

- *trackIndex* The track index of the cell (from 0 to 11).
- *columnIndex* The columnIndex index of the cell (from 0 to 3).
- *channel* the MIDI Channel, from 0 to 15 (MIDI channel is actually displayed in the WME Network screen as 1 to 16)

Example:

```
function onWmeStart() {  
  
    var synthFxNetworkIdForCellAtTrack0Column1MIDIChannel12 = waeGetFxNetworkIdForCellSynthTrackColumnChannel(0, 1, 12)  
  
}
```

### `waeGetFxNetworkIdForCellEffectTrackColumnChannel()`

This function returns the id that can be used for manipulating the effects chain for the Cell, identified by the supplied track and column indexes.

The function has the following parameters:

- *trackIndex* The track index of the cell (from 0 to 11).
- *columnIndex* The columnIndex index of the cell (from 0 to 3).
- *channel* the MIDI Channel, from 0 to 15 (MIDI channel is actually displayed in the WME Network screen as 1 to 16)

Example:

```
function onWmeStart() {  
  
    var trackFxNetworkIdForCellAtTrack0Column1MIDIChannel12  
  
}
```

#### `waeFxNetworkAllPresetNamesGet()`

This function returns an array of all FX presets that can be supplied to the FX network with the specified ID. The list returned using an FX Network ID generated by `waeGetFxNetworkIdForCellSynthTrackColumnChannel` is different to those using other FX network IDs (which all otherwise return the same array of values)

The function has the following parameters:

- *fxNetworkId* The id of the FX Network in question.

Example:

```
function onWmeStart() {  
  
    var fxNetworkIdForMix = waeGetFxNetworkIdForMix()  
    var allEffectsPresets = waeFxNetworkAllPresetNamesGet(  
  
    // Display the preset names on the console  
    for (var effectPresetIndex = 0; effectPresetIndex < all
```

```

    wjsLog(allEffectsPresets[effectPresetIndex])
}

var synthFxNetworkId = waeGetFxNetworkIdForCellSynthTrackColumnChannel
var allSynthPresets = waeFxNetworkAllPresetNamesGet(synthFxNetworkId)

// Display the preset names on the console
for (var synthFxIndex = 0; synthFxIndex < allSynthPresets.length; synthFxIndex++)
    wjsLog(allSynthPresets[synthFxIndex])
}
}

```

### `waeFxNetworkRandomPresetNamesGet()`

This function returns an array of all FX presets that can be supplied to the FX network with the specified ID; filtered such that only the presets that you have enabled in your current Schema (i.e. randomization schema) are returned. The list returned using an FX Network ID generated by `waeGetFxNetworkIdForCellSynthTrackColumnChannel` is different to those using other FX network IDs (which all otherwise return the same array of values)

The function has the following parameters:

- `fxNetworkId` The id of the FX Network in question.

Example:

```

function onWmeStart() {

    var fxNetworkIdForMix = waeGetFxNetworkIdForMix()
    var allEffectsPresets = waeFxNetworkRandomPresetNamesGet(fxNetworkIdForMix)

    // Display the preset names on the console
    for (var effectPresetIndex = 0; effectPresetIndex < allEffectsPresets.length; effectPresetIndex++)
        wjsLog(allEffectsPresets[effectPresetIndex])
}
}

```

```

}

var synthFxNetworkId = waeGetFxNetworkIdForCellSynthTra
var allSynthPresets = waeFxNetworkRandomPresetNamesGet

// Display the preset names on the console
for (var synthFxIndex = 0; synthFxIndex < allSynthPresets
    wjsLog(allSynthPresets.length[synthFxIndex])
}
}

```

### `waeFxNetworkPresetGet()`

This function returns the FX Preset for the specified FX Network.

Example:

```

function onWmeStart() {

    var fxNetworkIdForMix = waeGetFxNetworkIdForMix()
    waeFxNetworkPresetGet()
}

```

### `waeFxNetworkPresetSet()`

This function applies the specified FX Preset to the specified WAE FX Network. You can use a built-in Preset or a custom [WAE Synth Preset](#) or [WAE FX Preset](#) as relevant.

The function has the following parameters:

- *fxNetworkId* The id of the FX Network in question.
- *fxPreset* The preset to apply to the FX Network

Example:



```
function onWmeStart() {

    var fxNetworkIdForMix = waeGetFxNetworkIdForMix()

    // Built-in Preset
    waeFxNetworkPresetSet(fxNetworkIdForMix, "fx/Filter/Fi'

    // Custom Preset
    waeFxNetworkPresetSet(fxNetworkIdForMix, '<fxm><unit t=
}
```



### Synth Preset List

The following table all Synth Presets available to *waeFxNetworkPresetSet* in the current version of Wotja. The list was created using Wotja Script and the [waeFxNetworkAllPresetNamesGet](#) function (see the example code shown for that).

```
sounds/18 Bass/Bass fat swept.tg
sounds/18 Bass/Bass filter sweep.tg
sounds/18 Bass/Bass pluck soft.tg
sounds/18 Bass/Bass plucked bright.tg
sounds/18 Bass/Bass pulser.tg
sounds/18 Beats/Beats ch10 eperc wt.tg
sounds/18 Beats/Beats ch10 imdrums wt.tg
sounds/18 Beats/Beats ch10 ogglegacy wt.tg
sounds/18 Bell/Bell bright.tg
sounds/18 Bell/Bell combo.tg
sounds/18 Bell/Bell fs.tg
sounds/18 Bell/Bell glass felt.tg
sounds/18 Bell/Bell glass plastic.tg
sounds/18 Bell/Bell glass tri.tg
sounds/18 Bell/Bell synth bright.tg
sounds/18 Bell/Bell synth soft.tg
sounds/18 E-piano/E-Piano 2.tg
sounds/18 E-piano/E-Piano blend 2.tg
sounds/18 E-piano/E-Piano blend.tg
sounds/18 E-piano/E-Piano bowed.tg
sounds/18 E-piano/E-Piano bright.tg
sounds/18 E-piano/E-Piano shimmer.tg
sounds/18 E-piano/E-Piano soft.tg
sounds/18 E-piano/E-Piano space.tg
sounds/18 E-piano/E-Piano.tg
sounds/18 Lead/Lead bob.tg
sounds/18 Lead/Lead dsynth soft warm.tg
sounds/18 Multi/Multi 3 op fm template.tg
sounds/18 Multi/Multi piano strings.tg
sounds/18 Multi/Multi wavetable crossfade template.tg
```

sounds/18 Noise/Noise mains hum.tg  
sounds/18 Noise/Noise on the beach.tg  
sounds/18 Noise/Noise ring mod high bell.tg  
sounds/18 Noise/Noise tuned breezes.tg  
sounds/18 Pad/Pad bowed.tg  
sounds/18 Pad/Pad brass metal shimmer.tg  
sounds/18 Pad/Pad brass soft.tg  
sounds/18 Pad/Pad brassy.tg  
sounds/18 Pad/Pad coda mod.tg  
sounds/18 Pad/Pad epiano.tg  
sounds/18 Pad/Pad fizz grower.tg  
sounds/18 Pad/Pad fizz.tg  
sounds/18 Pad/Pad hummer.tg  
sounds/18 Pad/Pad outer space.tg  
sounds/18 Pad/Pad pipe shimmer.tg  
sounds/18 Pad/Pad pitched metal.tg  
sounds/18 Pad/Pad side trails vib.tg  
sounds/18 Pad/Pad soft chev 2.tg  
sounds/18 Pad/Pad soft easel.tg  
sounds/18 Pad/Pad soft ep 1.tg  
sounds/18 Pad/Pad soft ep 2.tg  
sounds/18 Pad/Pad soft for chording.tg  
sounds/18 Pad/Pad soft tremolo.tg  
sounds/18 Pad/Pad space park 1.tg  
sounds/18 Pad/Pad space park 2.tg  
sounds/18 Pad/Pad space park 3.tg  
sounds/18 Pad/Pad space scape.tg  
sounds/18 Pad/Pad space shimmer.tg  
sounds/18 Pad/Pad space wind.tg  
sounds/18 Pad/Pad synth 1.tg  
sounds/18 Pad/Pad synth 2.tg  
sounds/18 Pad/Pad unresolved.tg  
sounds/18 Pad/Pad warm chev 1.tg  
sounds/18 Pad/Pad warm soft.tg  
sounds/18 Perc/Perc beat sync noise.tg  
sounds/18 Piano/Piano blend bright.tg  
sounds/18 Piano/Piano blend upright.tg  
sounds/18 Piano/Piano brass.tg  
sounds/18 Piano/Piano damped bright.tg  
sounds/18 Piano/Piano damped long 1.tg  
sounds/18 Piano/Piano damped long 2.tg  
sounds/18 Piano/Piano damped.tg  
sounds/18 Piano/Piano upright long.tg  
sounds/18 Piano/Piano upright wide.tg  
sounds/18 Pluck/Pluck guitar bell bright.tg  
sounds/18 Pluck/Pluck guitar blend edge.tg  
sounds/18 Pluck/Pluck guitar blend ep.tg  
sounds/18 Pluck/Pluck guitar blend osc.tg  
sounds/18 Pluck/Pluck guitar blend soft.tg  
sounds/18 Pluck/Pluck guitar blend steel.tg  
sounds/18 Pluck/Pluck guitar bowed.tg  
sounds/18 Strings/Strings build.tg  
sounds/18 Strings/Strings distant.tg  
sounds/18 Strings/Strings slow.tg  
sounds/18 Strings/Strings tremolo.tg  
sounds/18 Synth/Synth beat sync filter blip.tg  
sounds/18 Synth/Synth dsynth warm 2.tg  
sounds/18 Synth/Synth fizz pulse.tg  
sounds/18 Synth/Synth fizz soft.tg  
sounds/18 Synth/Synth organ electric.tg

sounds/18 Synth/Synth pwm.tg  
sounds/18 Synth/Synth saw phase offset.tg  
sounds/18 Synth/Synth slow vowel morph.tg  
sounds/18 Synth/Synth soft lead.tg  
sounds/18 Synth/Synth subtractive template 1.tg  
sounds/18 Synth/Synth subtractive template.tg  
sounds/18 Synth/Synth swept harmonics.tg  
sounds/18 Synth/Synth wavetable simple vibrato.tg  
sounds/18 Units/Unit dsynth.tg  
sounds/18 Units/Unit osc.tg  
sounds/18 Units/Unit particle.tg  
sounds/18 Units/Unit wavetable.tg  
sounds/19 Drone/Drone aeolian spinner.tg  
sounds/19 Drone/Drone airier.tg  
sounds/19 Drone/Drone buzz chorus low.tg  
sounds/19 Drone/Drone buzz chorus sweep.tg  
sounds/19 Drone/Drone cosmic 1.tg  
sounds/19 Drone/Drone cosmic 2.tg  
sounds/19 Drone/Drone fizz 2.tg  
sounds/19 Drone/Drone fizz.tg  
sounds/19 Drone/Drone fm sweep variation.tg  
sounds/19 Drone/Drone fuzz.tg  
sounds/19 Drone/Drone gentle harmonic shifts.tg  
sounds/19 Drone/Drone gentle shifts pan.tg  
sounds/19 Drone/Drone index range.tg  
sounds/19 Drone/Drone lpf ebb blurred.tg  
sounds/19 Drone/Drone mild peril.tg  
sounds/19 Drone/Drone motor.tg  
sounds/19 Drone/Drone perpetual motion.tg  
sounds/19 Drone/Drone piano orch wash.tg  
sounds/19 Drone/Drone slow phase morph.tg  
sounds/19 Drone/Drone slow phase pan.tg  
sounds/19 Drone/Drone slow phase strings.tg  
sounds/19 Drone/Drone soft chorus sweep.tg  
sounds/19 Drone/Drone strings octave.tg  
sounds/19 Drone/Drone strings.tg  
sounds/19 Drone/Drone subtle offset pulse.tg  
sounds/19 Drone/Drone subtle pulse.tg  
sounds/19 Drone/Drone synth soft.tg  
sounds/19 Drone/Drone voxaa.tg  
sounds/19 Drone/Drone warm fizzer.tg  
sounds/19 Drone/Drone wide chorus low.tg  
sounds/19 Drone/Drone wide subtle additive.tg  
sounds/19 Drone/Drone wind in wire.tg  
sounds/19 Lead (M01)/Brassy.tg  
sounds/19 Lead (M01)/Breathy chime.tg  
sounds/19 Lead (M01)/Bright FM keys 1.tg  
sounds/19 Lead (M01)/Chime bar 1.tg  
sounds/19 Lead (M01)/FM keys 1.tg  
sounds/19 Lead (M01)/FM keys 2.tg  
sounds/19 Lead (M01)/FM keys 3.tg  
sounds/19 Lead (M01)/FM keys 4.tg  
sounds/19 Lead (M01)/FM keys 5.tg  
sounds/19 Lead (M01)/FM keys 6.tg  
sounds/19 Lead (M01)/FM pluck.tg  
sounds/19 Lead (M01)/Felt mallet chime.tg  
sounds/19 Lead (M01)/Filter warble.tg  
sounds/19 Lead (M01)/Harmonic scan 1.tg  
sounds/19 Lead (M01)/Inverse harmonic scan.tg  
sounds/19 Lead (M01)/Mellow FM keys 1.tg

sounds/19 Lead (M01)/Perc hi-lo.tg  
sounds/19 Lead (M01)/Pluck octaves.tg  
sounds/19 Lead (M01)/Rattle plate.tg  
sounds/19 Lead (M01)/Resonant filter sweep.tg  
sounds/19 Lead (M01)/Ringing sine.tg  
sounds/19 Lead (M01)/Slight vocal.tg  
sounds/19 Lead (M01)/Soft bright lead.tg  
sounds/19 Lead (M01)/Soft filter sweep.tg  
sounds/19 Lead (M01)/Soft hi-lo.tg  
sounds/19 Lead (M01)/Soft perc tine.tg  
sounds/19 Lead (M01)/Soft pluck bright tail.tg  
sounds/19 Lead (M01)/Staccato supersaws hard attack.tg  
sounds/19 Lead (M01)/Staccato supersaws.tg  
sounds/19 Lead (M01)/Supersaws hard attack.tg  
sounds/19 Lead (M01)/Supersaws.tg  
sounds/19 Lead (M01)/Tuned noise.tg  
sounds/19 Lead (M01)/WTable scan 1.tg  
sounds/19 Lead (M01)/WTable scan 2.tg  
sounds/19 Lead (M01)/WTable scan 3.tg  
sounds/19 Lead (M01)/WTable scan 4.tg  
sounds/19 Lead (M01)/WTable scan 5.tg  
sounds/19 Lead (M01)/WTable scan 6.tg  
sounds/24 Arp Lead/M02 4 pole gentle sweep.wotja\_sy  
sounds/24 Arp Lead/M02 bright pluck.wotja\_sy  
sounds/24 Arp Lead/M02 fm bell.wotja\_sy  
sounds/24 Arp Lead/M02 gentle filter arp.wotja\_sy  
sounds/24 Arp Lead/M02 pluck attack.wotja\_sy  
sounds/24 Arp Lead/Osc sweep 2 pole gentle.wotja\_sy  
sounds/24 Arp Lead/Osc sweep 4 pole gentle.wotja\_sy  
sounds/24 Arp Lead/Osc sweep 4 pole resonant.wotja\_sy  
sounds/24 Arp Lead/Res 0 glock morphing.wotja\_sy  
sounds/24 Arp Lead/Res 0 glock.wotja\_sy  
sounds/24 Arp Lead/Res 0 harmonic shifts.wotja\_sy  
sounds/24 Arp Lead/Res 0 long tail pluck.wotja\_sy  
sounds/24 Arp Lead/Res 0 marimbish.wotja\_sy  
sounds/24 Arp Lead/Res 0 metal overtones.wotja\_sy  
sounds/24 Arp Lead/Res 0 phasing metal.wotja\_sy  
sounds/24 Arp Lead/Res 0 strong metal.wotja\_sy  
sounds/24 Arp Lead/Res 2 acoustic guitar.wotja\_sy  
sounds/24 Arp Lead/Res 2 bang on a can can.wotja\_sy  
sounds/24 Arp Lead/Res 2 chorused electric guitar.wotja\_sy  
sounds/24 Arp Lead/Res 2 clean electric guitar.wotja\_sy  
sounds/24 Arp Lead/Res 2 electric piano.wotja\_sy  
sounds/24 Arp Lead/Res 2 soft bell.wotja\_sy  
sounds/24 Arp Lead/Res 2 tight pluck.wotja\_sy  
sounds/24 E-piano/E-Piano modern ep arp.wotja\_sy  
sounds/24 E-piano/E-Piano modern ep ove x2.wotja\_sy  
sounds/24 E-piano/E-Piano modern ep.wotja\_sy  
sounds/24 E-piano/E-Piano muted rhodes arp.wotja\_sy  
sounds/24 E-piano/E-Piano muted rhodes ove x2.wotja\_sy  
sounds/24 E-piano/E-Piano muted rhodes.wotja\_sy  
sounds/24 E-piano/E-Piano soft perc ep arp.wotja\_sy  
sounds/24 E-piano/E-Piano soft perc ep doubled.wotja\_sy  
sounds/24 E-piano/E-Piano soft perc ep ove x2.wotja\_sy  
sounds/24 E-piano/E-Piano soft perc ep.wotja\_sy  
sounds/24 E-piano/E-Piano vintage ep1 arp.wotja\_sy  
sounds/24 E-piano/E-Piano vintage ep1 doubled.wotja\_sy  
sounds/24 E-piano/E-Piano vintage ep1 ove x2.wotja\_sy  
sounds/24 E-piano/E-Piano vintage ep1.wotja\_sy  
sounds/24 E-piano/E-Piano vintage ep2 arp.wotja\_sy

sounds/24 E-piano/E-Piano vintage ep2 doubled.wotja\_sy  
sounds/24 E-piano/E-Piano vintage ep2 ove x2.wotja\_sy  
sounds/24 E-piano/E-Piano vintage ep2.wotja\_sy  
sounds/24 E-piano/E-piano modern ep doubled.wotja\_sy  
sounds/24 E-piano/E-piano muted rhodes doubled.wotja\_sy  
sounds/24 Piano/Piano blend ove res x3.wotja\_sy  
sounds/24 Piano/Piano blend ove x3.wotja\_sy  
sounds/24 Piano/Piano damped long ove x2.wotja\_sy  
sounds/24 Piano/Piano damped long.wotja\_sy  
sounds/24 Piano/Piano damped mid ove x2.wotja\_sy  
sounds/24 Piano/Piano damped mid.wotja\_sy  
sounds/24 Piano/Piano damped pizz.wotja\_sy  
sounds/24 Piano/Piano damped short ove x2.wotja\_sy  
sounds/24 Piano/Piano doubled.wotja\_sy  
sounds/24 Strings/Strings blend 5th x2.wotja\_sy  
sounds/24 Strings/Strings endless ove x2.wotja\_sy  
sounds/24 Strings/Strings endless x2.wotja\_sy  
sounds/24 Strings/Strings ensemble blend ove x3.wotja\_sy  
sounds/24 Strings/Strings ensemble ove x2.wotja\_sy  
sounds/24 Strings/Strings ensemble.wotja\_sy  
sounds/24 Strings/Strings pizz ove x2.wotja\_sy  
sounds/24 Strings/Strings pizz x2.wotja\_sy  
sounds/24 Strings/Strings synth ove x2.wotja\_sy  
sounds/24 Strings/Strings synth phasing x2.wotja\_sy  
sounds/24 Strings/Strings synth x2.wotja\_sy

## Effect Preset List

The following table all Effect Presets available to `waeFxNetworkPresetSet` in the current version of Wotja. The list was created using Wotja Script and the [waeFxNetworkAllPresetNamesGet](#) function (see the example code shown for that).

fx/18 Amp/- Amp default.fxm  
fx/18 Amp/Amp quad.fxm  
fx/18 Chorus/- Chorus default.fxm  
fx/18 Chorus/Chorus deep.fxm  
fx/18 Chorus/Chorus double.fxm  
fx/18 Chorus/Chorus fast flange.fxm  
fx/18 Chorus/Chorus feedback.fxm  
fx/18 Chorus/Chorus light.fxm  
fx/18 Chorus/Chorus medium.fxm  
fx/18 Chorus/Chorus strong.fxm  
fx/18 Compressor/- Compressor default.fxm  
fx/18 Compressor/Compressor basic.fxm  
fx/18 Compressor/Compressor low end boost.fxm  
fx/18 Compressor/Compressor mild.fxm  
fx/18 Compressor/Compressor smooth.fxm  
fx/18 Compressor/Compressor tight.fxm  
fx/18 Delay/- Delay default.fxm  
fx/18 Delay/Delay bounce.fxm  
fx/18 Delay/Delay double sync.fxm  
fx/18 Delay/Delay doubling.fxm  
fx/18 Delay/Delay echoer.fxm  
fx/18 Delay/Delay filtered mono.fxm  
fx/18 Delay/Delay filtered sync pan.fxm  
fx/18 Delay/Delay long.fxm  
fx/18 Delay/Delay resonator.fxm  
fx/18 Delay/Delay simple.fxm

fx/18 Delay/Delay slapback.fxm  
fx/18 Delay/Delay tight slapback.fxm  
fx/18 Distortion/- Distortion default.fxm  
fx/18 Distortion/Distortion broken filter.fxm  
fx/18 Distortion/Distortion fuzz drive.fxm  
fx/18 Distortion/Distortion fuzz.fxm  
fx/18 Distortion/Distortion mild.fxm  
fx/18 EQ/- Equaliser default.fxm  
fx/18 EQ/Equaliser hi cut.fxm  
fx/18 EQ/Equaliser lo cut.fxm  
fx/18 EQ/Equaliser mid.fxm  
fx/18 Filter/- Filter default.fxm  
fx/18 Filter/Filter 2Hz autowah.fxm  
fx/18 Filter/Filter 4Hz auto wah.fxm  
fx/18 Filter/Filter 4Hz autowah.fxm  
fx/18 Filter/Filter dirty.fxm  
fx/18 Filter/Filter hi cut.fxm  
fx/18 Filter/Filter high fast sweep.fxm  
fx/18 Filter/Filter parallel bandpass 2 bar sweep.fxm  
fx/18 Filter/Filter slow auto wah.fxm  
fx/18 Filter/Filter slow sweep hi q.fxm  
fx/18 Filter/Filter subtle sweep.fxm  
fx/18 Filter/Filter telephone.fxm  
fx/18 Overdrive/- Overdrive default.fxm  
fx/18 Overdrive/Overdrive extreme.fxm  
fx/18 Overdrive/Overdrive harmonics.fxm  
fx/18 Overdrive/Overdrive hot.fxm  
fx/18 Overdrive/Overdrive serious.fxm  
fx/18 Overdrive/Overdrive warm.fxm  
fx/18 Pan/Pan auto.fxm  
fx/18 Pan/Pan shimmer.fxm  
fx/18 Reverb/- Reverb default.fxm  
fx/18 Reverb/Reverb 5 second.fxm  
fx/18 Reverb/Reverb 9 second.fxm  
fx/18 Reverb/Reverb basement.fxm  
fx/18 Reverb/Reverb bright hall.fxm  
fx/18 Reverb/Reverb bright plate.fxm  
fx/18 Reverb/Reverb bright small room.fxm  
fx/18 Reverb/Reverb drum hallway.fxm  
fx/18 Reverb/Reverb drum studio.fxm  
fx/18 Reverb/Reverb light ambience.fxm  
fx/18 Reverb/Reverb medium room.fxm  
fx/18 Reverb/Reverb resonant metallic.fxm  
fx/18 Reverb/Reverb ringing 6 second.fxm  
fx/18 Reverb/Reverb slapback room.fxm  
fx/18 Reverb/Reverb smooth 10 second.fxm  
fx/18 Volume/Volume 10 second duck.fxm  
fx/18 Volume/Volume 4hz gate pan.fxm  
fx/18 Volume/Volume 4hz gate.fxm  
fx/18 Volume/Volume swell.fxm  
fx/18 Volume/Volume tremolo fast.fxm  
fx/18 Volume/Volume tremolo mono 2xbpm sync.fxm  
fx/18 Volume/Volume tremolo pan.fxm  
fx/18 Volume/Volume tremolo slow.fxm  
fx/18 Volume/Volume tremolo stereo 2xbpm sync.fxm  
fx/21 Mix FX/Combi chorus delay reverb 2.fxm  
fx/21 Mix FX/Combi chorus delay reverb.fxm  
fx/21 Mix FX/Combi chorus reverb.fxm  
fx/21 Mix FX/Combi compressor chorus delay eq.fxm  
fx/21 Mix FX/Combi compressor chorus delay reverb eq.fxm

fx/21 Mix FX/Combi delay chorus reverb.fxm  
fx/21 Mix FX/Combi delay eq.fxm  
fx/21 Mix FX/Combi delay reverb eq.fxm  
fx/21 Mix FX/Combi eq chorus reverb.fxm  
fx/21 Mix FX/Combi filter delay 2.fxm  
fx/21 Mix FX/Combi filter delay reverb muted.fxm  
fx/21 Mix FX/Combi filter delay shift.fxm  
fx/21 Mix FX/Combi filter delay.fxm  
fx/21 Mix FX/Combi filter eq reverb.fxm  
fx/21 Mix FX/Combi filter shifter.fxm  
fx/21 Mix FX/Combi lfo filter chorus reverb.fxm  
fx/21 Mix FX/Combi lfo filter chorus.fxm  
fx/21 Mix FX/Combi mix eq compressor reverb.fxm  
fx/21 Mix FX/Combi reverb eq.fxm  
fx/21 Mix FX/Combi reverb lfo filter.fxm  
fx/22 Ambience/Big reverb chorus (R1).fxm  
fx/22 Ambience/Big reverb chorus feedback (R1).fxm  
fx/22 Ambience/Big reverb chorus input pan (R1).fxm  
fx/22 Ambience/Big reverb dual pre (R1).fxm  
fx/22 Ambience/Big reverb hard edge (R1).fxm  
fx/22 Ambience/Big reverb haze edge (R2XL).fxm  
fx/22 Ambience/Big reverb haze pure (R2XL).fxm  
fx/22 Ambience/Big reverb input pan (R1).fxm  
fx/22 Ambience/Big reverb space (R2XL).fxm  
fx/22 Ambience/Big reverb tail filter bp (R2XL).fxm  
fx/22 Ambience/Big reverb tail filter lp (R2XL).fxm  
fx/22 Ambience/Big reverb tail filter notch (R2XL).fxm  
fx/22 Ambience/Big reverb undamped (R1).fxm  
fx/22 Ambience/Shimmer dual halo (R2XL).fxm  
fx/22 Ambience/Shimmer dual micro high (R2XL).fxm  
fx/22 Ambience/Shimmer dual mixed (R1).fxm  
fx/22 Ambience/Shimmer dual mixed (R2XL).fxm  
fx/22 Ambience/Shimmer dual octave (R1).fxm  
fx/22 Ambience/Shimmer dual octave (R2XL).fxm  
fx/22 Ambience/Shimmer fifth (R1).fxm  
fx/22 Ambience/Shimmer fifth (R2XL).fxm  
fx/22 Ambience/Shimmer micro (R1).fxm  
fx/22 Ambience/Shimmer micro (R2XL).fxm  
fx/22 Ambience/Shimmer micro lfo (R2XL).fxm  
fx/22 Ambience/Shimmer octave (R2XL).fxm  
fx/23 Reverb (R2XL)/ABC Reverb 1.wotja\_fx  
fx/23 Reverb (R2XL)/ABC Reverb 10.wotja\_fx  
fx/23 Reverb (R2XL)/ABC Reverb 2.wotja\_fx  
fx/23 Reverb (R2XL)/ABC Reverb 3.wotja\_fx  
fx/23 Reverb (R2XL)/ABC Reverb 4.wotja\_fx  
fx/23 Reverb (R2XL)/ABC Reverb 5.wotja\_fx  
fx/23 Reverb (R2XL)/ABC Reverb 6.wotja\_fx  
fx/23 Reverb (R2XL)/ABC Reverb 7.wotja\_fx  
fx/23 Reverb (R2XL)/ABC Reverb 8.wotja\_fx  
fx/23 Reverb (R2XL)/ABC Reverb 9.wotja\_fx  
fx/24 Combi/Combi delay R2XL pitchshift.wotja\_fx  
fx/24 Combi/Combi delay compressor.wotja\_fx  
fx/24 Combi/Combi dry R2XL EQ.wotja\_fx  
fx/24 Combi/Combi dry delay filter R1.wotja\_fx  
fx/24 Combi/Combi filtered delay.wotja\_fx  
fx/24 Combi/Combi sync delay filter.wotja\_fx  
fx/24 Reverb (R2XL)/Approaching the infinite.wotja\_fx  
fx/24 Reverb (R2XL)/Big tale.wotja\_fx  
fx/24 Reverb (R2XL)/Bow wave.wotja\_fx  
fx/24 Reverb (R2XL)/Clean haze.wotja\_fx

```
fx/24 Reverb (R2XL)/Home cooking.wotja_fx  
fx/24 Reverb (R2XL)/Plane sailing.wotja_fx  
fx/24 Reverb (R2XL)/Space field.wotja_fx  
fx/24 Reverb (R2XL)/Summer haze.wotja_fx
```

## WTE functions

WTE related functions all start with the `wte` prefix.

### `wteRandomWordsGet()`

This function returns a string containing a number of random words.

The function has the following parameters:

- `numberOfWords` The number of random words to return.

Example:

```
function onWmeStart() {  
  
    // Make a string with 3 random words  
    var ttmText = wteRandomWordsGet(3)  
  
    // Set the TTM text using our random string  
    wmeParameterSet("Text Pattern Text", ttmText)  
}
```

### `wteShuffleWords()`

This function returns a string containing a shuffled version of the supplied text. The words preserved from the supplied text, but the order of those words is changed.

The function has the following parameters:

- `textToShuffle` The text string to shuffle.

Example:



```
function onWmeStart() {  
  
    // Make a string that is a random shuffle of words  
    var textToShuffle = "Shuffle these four words"  
    var ttmText = wteShuffleWords(textToShuffle)  
  
    // Set the TTM text using our shuffled string  
    wmeParameterSet("Text Pattern Text", ttmText)  
}
```

### `wteShuffleCharacters()`

This function returns a string containing a shuffled version of the supplied text. All the characters are shuffled; words are broken-up.

The function has the following parameters:

- *textToShuffle* The text string to shuffle.

Example:

```
function onWmeStart() {  
  
    // Make a string that is a random shuffle of characters  
    var textToShuffle = "Shuffle these characters"  
    var ttmText = wteShuffleCharacters(textToShuffle)  
  
    // Set the TTM text using our shuffled string  
    wmeParameterSet("Text Pattern Text", ttmText)  
}
```



## <> Scripting Cookbook

### Trigger Script Cookbook

A great way to start thinking about Event Handler Script Functions, is to look at various real examples of how to do various interesting things with Event Handler Script Functions!

#### Example: How to choose a random Mix Root every time the mix starts

```
// Choose a random root

function chooseRandomRoot() {
    var roots = ["A", "A#", "B", "C", "C#", "D", "D#", "E",

    var rootIndex = wjsRandomGetFromTo(0, roots.length-1)
    var newRoot = roots[rootIndex]
    return newRoot
}

function onWmeStart() {
    wjsLog("onWmeStart!")

    var newRoot = chooseRandomRoot()
    wmeParameterSet("Mix Root", newRoot)
    wjsLog("Mix Root now=" + newRoot)
}
```



#### Example: Change the scale to use depending on time of day

This is a script that you could use as a cell-level *Bar* trigger script.

```
function onWmeStart() {
    var date = new Date()
```

```

var hour = date.getHours()
if ((hour > 20) || (hour <8)) {
    wmeParameterSet("Scale Rules", "early morning scale")
} else {
    wmeParameterSet("Scale Rules", "middle of day scale")
}
}

```

## Example: randomly select a pattern to use for a generator

This is a generator-level *Bar* trigger script.

```

function onWmeStart() {
    // Get a random integer between 1 and 3 inclusive.
    var value = wjsRandomGetFromTo (1, 3)

    wjsLog("value=" + value)

    var patternString = ""
    if (value == 1) {
        patternString = "<100 B 240 1>"
    } else if (value == 2) {
        patternString = "<100 B 120 1 120 2>"
    } else {
        patternString = "<100 B 60 1 60 2 60 3 60 4>"
    }

    wmeParameterSet("Patterns", patternString)
}

```

## Example: script trigger that allows Event Handler Script Functions to emit MIDI CC events, with an optional delay

This is a script that you could use as a cell-level *Bar* trigger script.

```
function onWmeBar(bar) {  
  
    // Generator trigger...  
    // Apply a pan sweep through the bar from left to right  
    // to show-off the use of wmeMIDISendCC.  
  
    wjsLog("onWmeBar(" + index + ") - Generator")  
  
    var barDuration = wmeBarDurationGet()  
  
    wjsLog("barDuration=" + barDuration)  
  
    var midiChannel = wmeGeneratorMIDIChannelGet()  
    var delay = 0  
    var ccController = 10 // Pan controller!  
    var value = 0  
    while (delay <= barDuration) {  
        var value = (127 * delay) / barDuration  
        // Note that the "delay" is optional; we're using the  
        // demo, to get a sweep effect from start to end of t  
        wmeMIDISendCC(midiChannel, ccController, value, delay  
        delay = delay + 20  
    }  
}
```

**Example: cell bar event trigger, that automatically adjusts the mix root and tempo every 2 bars**

```
function onWmeBar(bar) {
```

```

// Cell trigger...
// Change bar / tempo every 2 bars...

wjsLog("onWmeBar(" + bar + ") - Mix")

if ( ((bar + 1) % 2) == 0) {
    wjsLog("bar=" + bar)

    var rootWas = wmeParameterGet("Mix Root")

    wjsLog("Mix Root was=" + rootWas)

    // Every 2 bars, change the root at random,
    // from all those available!
    var roots = ["A", "A#", "B", "C", "C#", "D", "D#", "E

    var rootIndex = wjsRandomGetFromTo(0, roots.length-1)
    var newRoot = roots[rootIndex]
    wjsLog("New root=" + newRoot)

    wmeParameterSet("Mix Root", newRoot)

    // Set tempo as well; to a value related to the selec
    var tempoWas = wmeParameterGet("Tempo")
    wjsLog("Mix tempo was=" + tempoWas)

    var newTempo = 100 + rootIndex * 5
    wjsLog("New tempo=" + newTempo)
    wmeParameterSet("Tempo", newTempo)
}

```

```
}
```

## Example: cell bar event trigger, that automatically adjusts the Mix Root and Scale Rule every 2 bars

```
// Choose a random root
```

```
function chooseRandomRoot() {  
    var roots = ["A", "A#", "B", "C", "C#", "D", "D#", "E",  
  
    var rootIndex = wjsRandomGetFromTo(0, roots.length-1)  
    var newRoot = roots[rootIndex]  
    return newRoot  
}
```

```
// Check if the specified Scale Rule name exists in the m
```

```
function getDoesScaleRuleExistWithThisName(lookForRuleName) {  
    var scaleRuleCount = wmeObjectCountGet("Scale Rule")  
    //wjsLog("Scale Rules objects=" + scaleRuleCount)  
  
    for (var index = 0; index < scaleRuleCount; index++) {  
        var scaleRuleName = wmeObjectNameGet("Scale Rule", index)  
  
        // wjsLog("scaleRuleName=" + scaleRuleName)  
  
        if (lookForRuleName == scaleRuleName) {  
            // Found the rule, so return true.  
            return true  
        }  
    }  
}
```

```

// To get here, the rule wasn't found, so return false.

return false
}

// Event Triggers

function onWmeBar(bar) {

    // Cell trigger...
    // Change bar / scale rule every 2 bars...
    // Note that this requires you to have at least two sca
    // one called "Major" and one called "Minor"

    wjsLog("onWmeBar(" + bar + ") - Mix")

    // Every 2 bars, change the root at random, from all th
    if ( ((bar + 1) % 2) == 0) {
        wjsLog("bar=" + bar)

        var rootWas = wmeParameterGet("Mix Root")

        wjsLog("Mix Root was=" + rootWas)
        var newRoot = chooseRandomRoot()
        wmeParameterSet("Mix Root", newRoot)
        wjsLog("Mix Root now=" + newRoot)

        // Set scale as well; to a value related to the selec

        // Choose one of two Scale Rule names at random.

        var useRuleName = ""

```

```

if (wjsRandomGetFromTo(0, 100) <= 50) {
    useRuleName = "Major"
} else {
    useRuleName = "Minor"
}

if (getDoesScaleRuleExistWithThisName(useRuleName)) {
    wjsLog("Mix Scale Rule set to=" + useRuleName)

    // Apply rule to all generators!
    var generatorCount = wmeObjectCountGet("Generator")
    // wjsLog("Generator objects=" + generatorCount)

    for (var generatorIndex = 0; generatorIndex < gener
        // wjsLog("Generator name=" + wmeObjectNameGet("G
        wmeObjectParameterSet("Generator", generatorIndex
    }
} else {
    wjsLog("Warning - rule name not found=" + useRuleNa
}
}
}

```

**Example: cell start event trigger, that automatically adjusts the tempo to suit the time of day**

```

function onWmeStart() {
    wjsLog("Mix!")

    // Wotja script to get hour of day as 24-hour clock val
    // ... and adjust tempo accordingly!

```



```

var date = new Date()
var hour = date.getHours()
if ((hour > 20) || (hour <8)) {
    wjsLog("late night / early morning!")
    var randomValue = wjsRandomGetFromTo(0,50)
    wmeParameterSet("Tempo", 50 + randomValue)
} else {
    wjsLog("middle of day!")
    var randomValue = wjsRandomGetFromTo(0,50)
    wmeParameterSet("Tempo", 100 + randomValue)
}
}

```

### Example: generator start event trigger, that automatically cycles the Chord Depth every time it starts playback

```

function onWmeStart() {
    wjsLog("Generator!")

```

---

```

    var chordDepth = wmeParameterGet("Chord Depth")

    // Convert any string value to integer, then add one.
    chordDepth = parseInt(chordDepth) + 1

    // Loop the value around if it gets larger than we want
    if (chordDepth > 8) {
        chordDepth = 1
    }
    wmeParameterSet("Chord Depth", chordDepth)
}

```

---

## Example: generator event trigger, that automatically changes Chord Depth every bar

```
function onWmeBar(bar) {
  wjsLog("onWmeBar, bar=" + bar)
  var chordDepth = wmeParameterGet("Chord Depth")

  // Convert any string value to integer, then add one.
  chordDepth = parseInt(chordDepth) + 1

  // Loop the value around if it gets larger than we want
  if (chordDepth > 8) {
    chordDepth = 1
  }

  wmeParameterSet("Chord Depth", chordDepth)
}
```

## Example: bringing generators into the mix

In this example, imagine that you have a mutating drum Generator that you want to have muted at the start of the mix and come in with a bang at the start of bar 20. And you don't want the mutations to start until bar 21...

How would that work in the world of script?

Here is the Generator Start trigger:

```
function onWmeStart() {
  // When we start, disable mutation for this Generator!
  // And set mute, too!
  wmeParameterSet("Mute", "Yes")
  wmeParameterSet("Mutation Factor", 0)
```

```
}
```

Here is the Generator Bar trigger:

```
function onWmeBar(bar) {  
  if (bar == 20) {  
    // At bar 20, unmute the generator!  
    wmeParameterSet("Mute", "No")  
  }  
  
  if (bar == 21) {  
    // At bar 21, adjust the mutation factor!  
    wmeParameterSet("Mutation Factor", 20)  
  }  
}
```

The power of using scripts like this, is that:

- The drums always start on the bar specified.
- They don't morph until they are audible.
- The drum elements can now make staggered entry despite all being on the same MIDI channel.

## Example: Emit MIDI CC in response to composed note events

```
var GLastNote = null
```

```
function onWmeGeneratorComposed(generatorIndex, noteon, c  
  // Keep track of last composed note using the global (f  
  // called GLastNote ...  
  // Note that the "pitch" parameter should be ignored wh
```

```

midiChannel = wmeGeneratorMIDIChannelGet()

if (noteon) {
    if (GLastNote != null) {
        wmeMIDISendCC(midiChannel, 11, GLastNote, 10)
    }

    GLastNote = pitch
    wmeMIDISendCC(midiChannel, 11, pitch, 0)
} else {
    if (GLastNote != null) {
        wmeMIDISendCC(midiChannel, 11, GLastNote, 10)
        GLastNote = null
    }
}
}

```

### Example: Change TTM Text for Generator At Start of Mix

```

function makeRandomText() {
    // Make some random TTM text.
    // We show 3 different techniques!
    // Choose one of these to use at random.
    var technique = wjsRandomGetFromTo(0, 2)
    var ttmText = ""

    switch (technique) {
    case 0:
        // Make a string with 3 random words
        ttmText = wteRandomWordsGet(3)
        break
    }
}

```

```

case 1:
    // Make a string that is a random shuffle of words
    ttmText = "Shuffle these four words"
    ttmText = wteShuffleWords(ttmText)
    break
default:
    // Make a string that is a random shuffle of characters
    ttmText = "Shuffle these characters"
    ttmText = wteShuffleCharacters(ttmText)
    break
}

wjsLog("random ttmText=", ttmText)

return ttmText
}

function onWmeStart() {

    // Generate some random TTM text!
    var ttmText = makeRandomText()

    // Now that we have our string, set it!
    wmeParameterSet("Text Pattern Text", ttmText)
}

```

### **Example: Change TTM Repeats at Start**

```

function onWmeStart() {
    // Set the TTM property "Text Pattern Repeats" to 2

```

```
wmeParameterSet("Text Pattern Repeats", 2)
}
```

## Example: Change TTM Text At Start Then Restore After 4 bars

```
// Complex scripting example:
// - At mix start, apply random TTM Text.
// - After 4 bars, or at mix stop - restore the original

// The original TTM text is stored in this "global" variable
var GOriginalTTMText = ""

// The value we last set in onWmeStart
var GLastSetTTMText = ""

function setRandomTTMText() {
  // Take a copy of the original TTM text
  GOriginalTTMText = wmeParameterGet("Text Pattern Text")
  wjsLog("GOriginalTTMText=" + GOriginalTTMText)

  // Make a string with 3 random words
  var ttmText = wteRandomWordsGet(3)
  wjsLog("ttmText=" + ttmText)

  GLastSetTTMText = ttmText

  // Now that we have our string, set it!
  wmeParameterSet("Text Pattern Text", ttmText)
}

function restoreOriginalTTMText() {
```

```

var ttmNow = wmeParameterGet("Text Pattern Text")
wjsLog("restoreOriginalTTMText, ttmNow=" + ttmNow)

if (ttmNow != GLastSetTTMText) {
    wjsLog("The user value has changed since last random
} else {
    wjsLog("Restoring TTM Text to=" + GOriginalTTMText)
    wmeParameterSet("Text Pattern Text", GOriginalTTMText
}
}

//
// Trigger script functions
//

function onWmeStart() {

    wjsLog("onWmeStart")

    setRandomTTMText()
}

function onWmeBar(bar) {
    // On the 4th bar, restore the TTM text... UNLESS the u
    // has already changed the value through the UI.

    wjsLog("onWmeBar, bar=" + bar)

    if (bar == 4) {

        wjsLog("onWmeBar, bar 4!")
    }
}

```

```

    restoreOriginalTTMText()
}
}

function onWmeStop() {
    wjsLog("onWmeStop")
    // At the end, restore the TTM text... UNLESS the user
    // has already changed the value through the UI.
    restoreOriginalTTMText()
}

```

### Example: Cell: Query All Synth and Track Presets at Start

```

function onWmeStart() {

    // Query what Synth and Effect presets are available, a

    var trackIndex = 0
    var columnIndex = 0
    var channel = 1

    var fxNetworkIdSynth = waeGetFxNetworkIdForCellSynthTra
    var fxNetworkIdTrack = waeGetFxNetworkIdForCellEffectTr

    var synthFxPresets = waeFxNetworkAllPresetNamesGet(fxNe
    wjsLog("All synthFxPresets.length=" + synthFxPresets.le

    for (var synthFxIndex = 0; synthFxIndex < synthFxPreset
        wjsLog(synthFxPresets[synthFxIndex])
    }
}

```



```

var trackFxPresets = waeFxNetworkAllPresetNamesGet(fxNe
wjsLog("All trackFxPresets.length=" + trackFxPresets.le

for (var trackFxIndex = 0; trackFxIndex < trackFxPreset
    wjsLog(trackFxPresets[trackFxIndex])
}
}

```

## Example: Cell: Query And Set Random Effects and Patches

```

function onWmeStart() {

    // Generate a set of FX Network IDs, that are used for
    var fxNetworkIdMix = waeGetFxNetworkIdForMix()
    var fxNetworkIdMixTrack0 = waeGetFxNetworkIdForMixTrack
    var fxNetworkIdCellGlobalTrack0Column0 = waeGetFxNetwor
    var fxNetworkIdCellSynthTrack0Column0Channel1 = waeGetF
    var fxNetworkIdCellEffectTrack0Column0Channel1 = waeGet

    // Query what Synth and Effect presets are available, a

    var trackFxPresets = waeFxNetworkRandomPresetNamesGet(f
    wjsLog("trackFxPresets.length=" + trackFxPresets.length

    for (var trackFxIndex = 0; trackFxIndex < trackFxPreset
        wjsLog("trackFxPreset=" + trackFxPresets[trackFxIndex
    }

    var synthFxPresets = waeFxNetworkRandomPresetNamesGet(f
    wjsLog("synthFxPresets.length=" + synthFxPresets.length

```

```

for (var synthFxIndex = 0; synthFxIndex < synthFxPreset
    wjsLog("synthFxPreset=" + synthFxPresets[synthFxIndex
}

if (synthFxPresets.length > 0) {
    // Set a random synth patch (from those available) fo
    // Now... set patch random for cell 1!
    var synthPresetForTrack0Column0Channel1 = synthFxPres
    wjsLog("Use synthPresetForTrack0Column0Channel1=" + s

    waeFxNetworkPresetSet(fxNetworkIdCellSynthTrack0Colum
}

if (trackFxPresets.length > 0) {
    // Set a random FX patch (from those available) for t
    var fxForMix = trackFxPresets[wjsRandomGetFromTo(0, tr
    wjsLog("Use fxForMix=" + fxForMix)
    waeFxNetworkPresetSet(fxNetworkIdMix, fxForMix)
}
}

// Example Code to set the Rhythm Rule used by a Generato

    mixCellObjectParameterSet(columnIndex, trackIndex, "Gen

// Example Code to set the Scale Rule used by a Generator

mixCellObjectParameterSet(columnIndex, trackIndex, "Gener

// Example Code to replace a Generator's FX Network

```

```

// You first have to know which MIDI track the generator
var midiChannel = mixCellObjectParameterGet(columnIndex,
if (midiChannel >= 1) {
    // Convert from 1 to 16 to 0 to 15
    midiChannel = midiChannel - 1
}
//then set the value with:
var fxNetworkId = waeGetFxNetworkIdForCellEffectTrackColu
waeFxNetworkPresetSet(fxNetworkId, "fx/Filter/Filter 4Hz
// or
var fxNetworkId = waeGetFxNetworkIdForCellEffectTrackColu
waeFxNetworkPresetSet(fxNetworkId, '<fxm><unit t="filter"

// Example Code to replace the Mix FX Network
var fxNetworkId = waeGetFxNetworkIdForMix()
// Example Usage with Paths:
waeFxNetworkPresetSet(fxNetworkId, "fx/Filter/Filter 4Hz
// or using FX string directly:
var fxNetworkId = waeGetFxNetworkIdForMix()
waeFxNetworkPresetSet(fxNetworkId, '<fxm><unit t="filter"

// Example Code to replace the Template in the Cell

// Example Usage with Paths:
mixCellTemplateSet(columnIndex, trackIndex, "intermorphic
mixCellTemplateSet(0, 0, "intermorphic/mixtikl/paks/BP22

// Shorter alternative to the above (easier to read):
mixCellTemplateSet(columnIndex, trackIndex, "im/IM_ZG_Mor
mixCellTemplateSet(0, 0, "im/BP22 Drums/Clave and toms.no

// Alternatives using template URL:
mixCellTemplateSet(columnIndex, trackIndex, "wotja://temp

```

```

mixCellTemplateSet(columnIndex, trackIndex, "noatikl://..

// Example Code to replace a Generator's Synth Network

// You first have to know which MIDI track the generator
var midiChannel = mixCellObjectParameterGet(columnIndex,
if (midiChannel >= 1) {
    // Convert from 1 to 16 to 0 to 15
    midiChannel = midiChannel - 1
}
//then set the value with:
var fxNetworkId = waeGetFxNetworkIdForCellSynthTrackColum
waeFxNetworkPresetSet(fxNetworkId, "sounds/18 Bell/Bell f
// or
var fxNetworkId = waeGetFxNetworkIdForCellSynthTrackColum
waeFxNetworkPresetSet(fxNetworkId, '<fxm><unit t="tg" p="

```

### Example: Cell: Query And Set Random Global Preset at Start

```

function onWmeStart() {

    // Generate the FX Network ID ... for the Global Mix FX
    var fxNetworkIdMix = waeGetFxNetworkIdForMix()

    // Query what Effect presets are available... for the G
    var trackFxPresets = waeFxNetworkRandomPresetNamesGet(f

    // Set a random FX patch (from those available) for the

```

```

// Check - has the user got any suitable FX enabled?
if (trackFxPresets.length > 1) {
    // Yes, we have at least one suitable FX preset avail

    // Choose one at random...
    var randomIndex = wjsRandomGetFromTo(0,trackFxPresets
    var fxForMix = trackFxPresets[randomIndex]

    wjsLog("Use fxForMix=" + fxForMix)

    // Apply the randomly selected preset
    waeFxNetworkPresetSet(fxNetworkIdMix, fxForMix)
}
}

```

## Example: Generator Query And Set Random Synth and Track Preset at Start

```

function onWmeStart() {

    // Generator trigger script - apply a random Synth patc

    // Generate the FX Network ID ... for the Generator's c
    <
    var trackIndex = wmeCellTrackIndexGet()
    var columnIndex = wmeCellColumnIndexGet()
    var channel = wmeGeneratorMIDIChannelGet()
    wjsLog("Generator trackIndex=" + trackIndex + ", column

    var fxNetworkIdSynth = waeGetFxNetworkIdForCellSynthTra
    var fxNetworkIdTrack = waeGetFxNetworkIdForCellEffectTr

```

```

// Query what Synth and Effect presets are available, a
var synthFxPresets = waeFxNetworkRandomPresetNamesGet(f
wjsLog("synthFxPresets.length=" + synthFxPresets.length

for (var synthFxIndex = 0; synthFxIndex < synthFxPreset
  wjsLog("synthFxPreset=" + synthFxPresets[synthFxIndex
}

var trackFxPresets = waeFxNetworkRandomPresetNamesGet(f
wjsLog("trackFxPresets.length=" + trackFxPresets.length

for (var trackFxIndex = 0; trackFxIndex < trackFxPreset
  wjsLog("trackFxPreset=" + trackFxPresets[trackFxIndex
}

if (synthFxPresets.length > 0) {
  // Set a random synth patch (from those available) fo
  // Now... set patch random for cell 1!
  var synthPreset = synthFxPresets[wjsRandomGetFromTo(0
  wjsLog("Use synthPreset=" + synthPreset)

  waeFxNetworkPresetSet(fxNetworkIdSynth, synthPreset)
}

if (trackFxPresets.length > 0) {
  // Set a random FX patch (from those available) for t
  var trackPreset = trackFxPresets[wjsRandomGetFromTo(0
  wjsLog("Use fxForMix=" + trackPreset)
  waeFxNetworkPresetSet(fxNetworkIdTrack, trackPreset)
}
}

```

## Example: Mix Level script, showing various functions

```
// Scripting API methods related mix-level functions -

function mixObjectParameterGetAndSet() {
    // mixObjectParameterGet
    // mixObjectParameterSet

    // Example usage:
    var scaleRuleValue = mixObjectParameterGet("Scale Rule")
    wjsLog("Mix scale rule index 0, value was = " + scaleRuleValue)
    scaleRuleValue = mixObjectParameterSet("Scale Rule", scaleRuleValue)
    scaleRuleValue = mixObjectParameterGet("Scale Rule", scaleRuleValue)
    wjsLog("Mix scale rule index 0, value now = " + scaleRuleValue)
}

function mixObjectQuerying() {
    // mixObjectNameGet
    // mixObjectIndexGet
    // mixObjectCountGet

    // Example usage:
    var scaleRuleCount = mixObjectCountGet("Scale Rule")
    wjsLog("Mix scale rule count = " + scaleRuleCount)
    var scaleRuleName = mixObjectNameGet("Scale Rule", 0)
    wjsLog("Mix scale rule 0 has name = " + scaleRuleName)
    var indexofScaleRuleName = mixObjectIndexGet("Scale Rule", scaleRuleName)
    wjsLog("Mix scale rule with name=" + scaleRuleName + " has index=" + indexofScaleRuleName)
}
```

```

function mixCellObjectQuerying() {
    // mixCellObjectNameGet
    // mixCellObjectIndexGet
    // mixCellObjectCountGet

    // Example usage:
    var columnIndex = 0
    var trackIndex = 0
    var scaleRuleCount = mixCellObjectCountGet(columnIndex, trackIndex)
    wjsLog("Mix cell(" + columnIndex + "," + trackIndex + ") count = " + scaleRuleCount)
    var scaleRuleName = mixCellObjectNameGet(columnIndex, trackIndex)
    wjsLog("Mix cell(" + columnIndex + "," + trackIndex + ") name = " + scaleRuleName)
    var indexOfScaleRuleName = mixCellObjectIndexGet(columnIndex, trackIndex, scaleRuleName)
    wjsLog("Mix cell(" + columnIndex + "," + trackIndex + ") index of " + scaleRuleName + " = " + indexOfScaleRuleName)
}

```

```

function mixCellParameterGetAndSet() {
    // mixCellObjectParameterSet
    // mixCellObjectParameterGet

    // Example usage:
    var columnIndex = 0
    var trackIndex = 0
    var generatorIndex = 0
    var generatorValue = mixCellObjectParameterGet(columnIndex, trackIndex, generatorIndex)
    wjsLog("generatorValue = " + generatorValue)
    mixCellObjectParameterSet(columnIndex, trackIndex, generatorIndex, "G")
    generatorValue = mixCellObjectParameterGet(columnIndex, trackIndex, generatorIndex)
    wjsLog("generatorValue = " + generatorValue)
    var cellValue = mixCellObjectParameterGet(columnIndex, trackIndex, generatorIndex, "C")
    wjsLog("cellValue = " + cellValue)
    mixCellObjectParameterSet(columnIndex, trackIndex, generatorIndex, "C")
    cellValue = mixCellObjectParameterGet(columnIndex, trackIndex, generatorIndex, "C")
}

```



```

    wjsLog("cellValue = " + cellValue)
}

function mixParameterGetAndSet() {
    // mixParameterSet
    // mixParameterGet

    // Example usage:
    var mixValue = mixParameterGet("Scale Rules")
    wjsLog("Mix: Scale Rules = " + mixValue)
    mixParameterSet("Scale Rules", "Default")
    mixValue = mixParameterGet("Scale Rules")
    wjsLog("Mix: Scale Rules = " + mixValue)

    mixValue = mixParameterGet("Mix Root")
    wjsLog("Mix: Mix Root = " + mixValue)
    mixParameterSet("Mix Root", "C")
    mixValue = mixParameterGet("Mix Root")
    wjsLog("Mix: Mix Root = " + mixValue)

    mixValue = mixParameterGet("Tempo")
    wjsLog("Mix: Tempo = " + mixValue)
    mixParameterSet("Tempo", "100")
    mixValue = mixParameterGet("Tempo")
    wjsLog("Mix: Tempo = " + mixValue)
}

function mixVolumeGetAndSet() {
    // mixVolumeSet
    // mixVolumeGet

    // Example usage:
    var value = mixVolumeGet() // Get current value, from

```

```

print(value)
mixVolumeSet(0) // Set to minimum volume
value = mixVolumeGet()
print(value)
mixVolumeSet(127) // Set to maximum volume
value = mixVolumeGet()
print(value)
}

```

```

function mixTrackVolumeGetAndSet() {
  // mixTrackVolumeSet
  // mixTrackVolumeGet

  // Example usage:
  var trackIndex = 0
  var value = mixTrackVolumeGet(trackIndex) // Value fr
  wjsLog("Track volume = " + value)
  mixTrackVolumeSet(trackIndex, 100) // Value from 0 to
  value = mixTrackVolumeGet(trackIndex) // Value from 0
  wjsLog("Track volume = " + value)
}

```

```

function mixTrackPanGetAndSet() {
  // mixTrackPanSet
  // mixTrackPanGet

  // Example usage:
  var trackIndex = 0
  var value = mixTrackPanGet(trackIndex) // Value from
  wjsLog("Track pan = " + value)
  mixTrackPanSet(trackIndex, 64) // Value from 0 to 127
  value = mixTrackPanGet(trackIndex) // Value from 0 to
  wjsLog("Track pan = " + value)
}

```

```
}
```

## Example: Mix Level script, showing various functions and triggers

```
//  
// Example of script functions and triggers to cycle th  
// through the progression used in Chopin's 24 preludes  
//  
  
const cAllScaleRules = {  
  
  "Major":  
    "1 0 0.5 0 1 0.5 0 1 0 0.5 0 0.5",  
  "Minor":  
    "1 0 1 1 0 1 0 1 1 0 1 0",  
  "All Scale Major":  
    "1 0 1 0 1 1 0 1 0 1 0 1",  
  "All Scale Minor Natural":  
    "1 0 1 1 0 1 0 1 1 0 1 0",  
  "All Scale Minor Harmonic":  
    "1 0 1 1 0 1 0 1 0 0 0 1",  
  "Chord Major triad":  
    "1 0 0 0 1 0 0 1 0 0 0 0",  
  "Chord Major sixth":  
    "1 0 0 0 1 0 0 1 0 1 0 0",  
  "Chord Dominant seventh":  
    "1 0 0 0 1 0 0 1 0 0 1 0",  
  "Chord Major seventh":  
    "1 0 0 0 1 0 0 1 0 0 0 1",  
  "Chord Augmented triad":  
    "1 0 0 0 1 0 0 0 1 0 0 0",  
  "Chord Augmented seventh":
```

"1 0 0 0 1 0 0 0 1 0 1 0",  
 "Chord Minor triad":  
 "1 0 0 1 0 0 0 1 0 0 0 0",  
 "Chord Minor sixth":  
 "1 0 0 1 0 0 0 1 0 1 0 0",  
 "Chord Minor seventh":  
 "1 0 0 1 0 0 0 1 0 0 1 0",  
 "Chord Minor-major seventh":  
 "1 0 0 1 0 0 0 1 0 0 0 1",  
 "Chord Diminished triad":  
 "1 0 0 1 0 0 1 0 0 0 0 0",  
 "Chord Diminished seventh":  
 "1 0 0 1 0 0 1 0 0 1 0 0",  
 "Chord Half-diminished seventh":  
 "1 0 0 1 0 0 1 0 0 0 1 0",  
 "Chord Augmented major seventh":  
 "1 0 0 0 1 0 0 0 1 0 0 1",  
 "Chord Seven-six":  
 "1 0 0 0 1 0 0 1 0 0 1 0",  
 "Chord Mixed-third":  
 "1 0 0 1 1 0 0 1 0 0 0 0",  
 "Chord Suspended fourth":  
 "1 0 0 0 0 1 0 1 0 0 0 0",  
 "Chord Dominant ninth":  
 "1 0 0 0 1 0 0 1 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0",  
 "Chord Dominant eleventh":  
 "1 0 0 0 1 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0",  
 "Chord Dominant thirteenth":  
 "1 0 0 0 1 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 0 0 1 0 0 0 0 0",  
 "Chord Seventh minor ninth":  
 "1 0 0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0",  
 "Chord Seventh sharp ninth":  
 "1 0 0 0 1 0 0 1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0"

```

"Chord Seventh augmented eleventh":
  "1 0 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0"
"Chord Seventh diminished thirteenth":
  "1 0 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 1 0 0 0 0"
"Chord Add nine":
  "1 0 0 0 1 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0"
"Chord Add fourth":
  "1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0"
"Chord Add sixth":
  "1 0 0 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0"
"Chord Six-nine":
  "1 0 0 0 1 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0"
"Chord Suspended second":
  "1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0"
"Chord Jazz sus":
  "1 0 0 0 0 1 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0"
"Mode Ionian (I)":
  "1 0 1 0 1 1 0 1 0 1 0 1",
"Mode Dorian (II)":
  "1 0 1 1 0 1 0 1 0 1 1 0",
"Mode Phrygian (III)":
  "1 1 0 1 0 1 0 1 1 0 1 0",
"Mode Lydian (IV)":
  "1 0 1 0 1 0 1 1 0 1 0 1",
"Mode Mixolydian (V)":
  "1 0 1 0 1 1 0 1 0 1 1 0",
"Mode Aeolian (VI)":
  "1 0 1 1 0 1 0 1 1 0 1 0",
"Mode Locrian (VII)":
  "1 1 0 1 0 1 1 0 1 0 1 0"
}

```

```
const cProgression = [
```

```
"C Major",  
"A Minor",  
"G Major",  
"E Minor",  
"D Major",  
"B Minor",  
"A Major",  
"F# Minor",  
"E Major",  
"C# Minor",  
"B Major",  
"G# Minor",  
"F# Major",  
"Eb Minor",  
"Db Major",  
"Bb Minor",  
"Ab Major",  
"F Minor",  
"Eb Major",  
"C Minor",  
"Bb Major",  
"G Minor",  
"F Major",  
"D Minor"]
```

```
const cChangeEveryBars = 4
```

```
var cycleIndex = 0; // We cycle round from 0 through to
```

```
function applyScaleName(scaleName) {
```

```
    var newRuleValue = cAllScaleRules[scaleName]
```

```
    mixObjectParameterSet("Scale Rule", 0, "Value", newRu
```

```
var newRuleValue = mixObjectParameterGet("Scale Rule"  
wjsLog("Scale Rule Value (" + scaleName + "(, now=" +  
}
```

```
function applyRandomTempo() {  
    var newTempo = 60 + wjsRandomGetFromTo(0, 60)  
  
    mixParameterSet("Tempo", newTempo)  
  
    var testTempo = mixParameterGet("Tempo")  
    wjsLog("Mix tempo=" + testTempo)  
}
```

```
function applyProgression() {  
  
    var item = cProgression[cycleIndex]  
  
    // Split into Root, and scale  
    var pairOfItems = item.split(" ");  
  
    var newRoot = pairOfItems[0]  
  
    mixParameterSet("Mix Root", newRoot)  
    var testRoot = mixParameterGet("Mix Root")  
    wjsLog("Mix Root now=" + testRoot)  
  
    // Set scale as well; to a value related to the selec  
  
    var scaleName = pairOfItems[1]  
    applyScaleName(scaleName)  
  
    applyRandomTempo()
```

```

}

function startProgression() {
    cycleIndex = 0

    applyProgression()
}

function nextProgression() {
    cycleIndex = cycleIndex + 1
    if (cycleIndex >= cProgression.length) {
        // We've reached the end - go back to the beginning
        cycleIndex = 0
    }

    applyProgression()
}

// Check if the specified Scale Rule name exists in the
function checkForChangeInProgression(bar) {

    // Every 4 bars (or whatever...), change the root and
    // seen in Chopin's preludes...

    if ( ((bar + 1) % cChangeEveryBars) == 0) {

        nextProgression()
    }
}

// Event Triggers

function onWmeStart() {

```



```
    startProgression()
}

function onWmeBar(bar) {

    wjsLog("Mix: onWmeBar(" + bar + ") - Cell")
    checkForChangeInProgression(bar)
}
```

## Hyperinstrument

## What is a hyperinstrument?

A hyperinstrument can be defined as being an instrument that can have extraordinary, semi-automated response to relatively simple real-time inputs.

Wotja is a hyperinstrument in many ways. For example from simple interaction in real-time through mouse and keyboard, you can have it generate extraordinary, ever-changing music that responds to your inputs. Wotja also directly responds to any real-time MIDI note data fed into it, for example from a MIDI keyboard or other MIDI controller; where you feed MIDI data into Wotja, you will find that Wotja can automatically harmonize its generators with all incoming MIDI note data. You can also configure Wotja to respond in many advanced ways to both MIDI note events and MIDI control events (otherwise known as "*MIDI CCs*").

Wotja can function as a hyperinstrument for 2 main reasons: The WME can real-time harmonies with incoming MIDI note events and control events (otherwise known as "*MIDI CCs*"), and Wotja's generative engine is an instrument that creates more than the sum of your direct inputs with keyboard, mouse, and MIDI. In other words, Wotja's generative engine, MIDI control, scripting and direct MIDI input harmonization all work together to make itself greater than the sum of its parts; a **customizable generative hyperinstrument!**

See the [Hyperinstrument - Quick Start](#) if you want to get started quickly.

## How does Wotja work as a hyperinstrument?

You can respond to any [incoming MIDI CC](#) however you want; for example, to change the current scale or adjust a harmony rule in real-time in response to incoming MIDI CC events.

You can also respond to any [incoming MIDI Note on/off event](#) however you want; for example, you could use some notes (such as accidentals in a scale) to change parameter values. In other words, you can use certain specific notes to trigger various complex responses by Wotja, for example; or to enable/disable the chording parameters only for certain notes (immediately that a note is triggered!); or to change scale; or pretty much whatever you can imagine.

Here is an example of a Generator MIDI In Note trigger script that you could use in your own mix.

```
function onMIDIInNote(channel, noteon, pitch, velocity) {
  if (channel == wmeGeneratorMIDIChannelGet()) {
    wjsLog("onMIDIInNote for our channel!")
  }
}
```

## Quick Start - Listening generators / Hyperinstrument

To have a Generator "listen" to incoming MIDI data, you need to follow the instructions below.


1. Set "MIDI Input" to be your MIDI input device.
2. Set the generator you want to listen to MIDI input, such that:
  - the Generator Type is Listen
  - the MIDI In Script is something like this:

```
function onMIDIInNote(channel, noteon, pitch, velocity)
  if (channel == wmeGeneratorMIDIChannelGet()) {
    wjsLog("onMIDIInNote for our channel!")
  }
}
```

3. You might want to set your Listening generator to auto-chord with depth of 2 or greater.
4. If you want an different Generator to also make sounds in response to your incoming MIDI note event, set it to FOLLOW the Listening Generator.

© 1990-2024 Intermorphic Ltd. All Rights Reserved. Subject to Change. E & OE. See also [Credits](#).



Intermorphic  is a brother team with a 30+ year passion for developing powerful [generative music](#) apps, engines & tools.

We ❤️ our customers and ❤️ making apps to help aid creativity & relaxation.

**Wotja**

App

**Wotja Docs**

Wotja User Guide

[Downloads](#)

[Help & FAQs](#)

[Tutorials](#)

[Add-on Paks](#)

[Music Engine \(WME\)](#)

[Audio Engine \(WAE\)](#)

[Script Engine \(WSE\)](#)

[Wotja Pak Maker](#)

## News & Blog

[News](#)

[Blog > Posts](#)

[Press > Releases](#)

[Press > Resources](#)

[📁 News Archive !\[\]\(626ce8ac21792b9405bfddfea8e0c96a\_img.jpg\)](#)

## Other


[Contact Us](#)

[T-Shirts etc.](#)

[About Us !\[\]\(c1168d6a8b365d11e842ece304635fa7\_img.jpg\)](#)

[SSEYO Koan !\[\]\(cbd8541a32dfc32f356f5c6c994b0a21\_img.jpg\)](#)

[📁 App Archive !\[\]\(d3e32d099174a7c248ec1f564ee4f69c\_img.jpg\)](#)

© 1990-2024 Intermorphic . All rights reserved.

[Privacy Policy](#) | [Terms of Use](#) | [Wotja EULA](#) | [Sitemap](#)

