Learning DigiShow

8

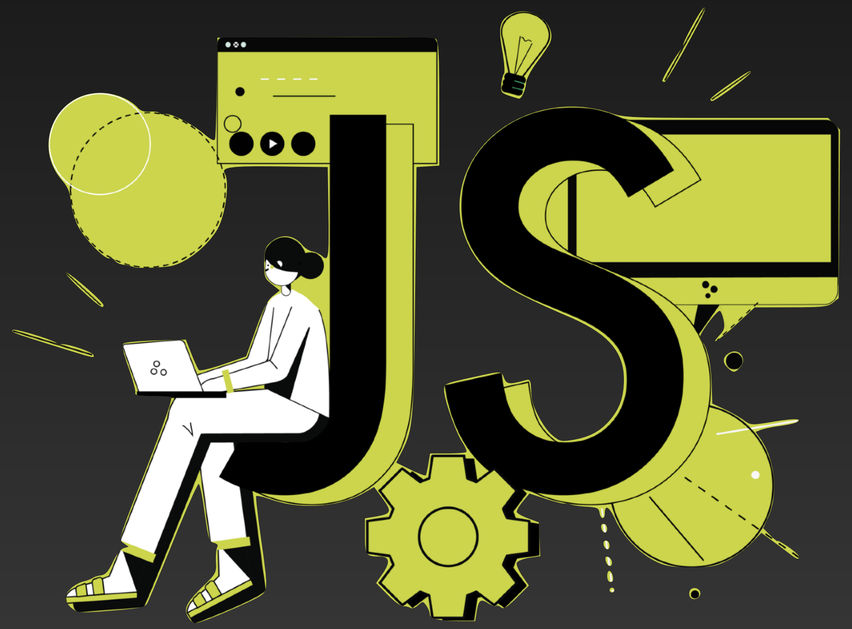
Expressions and Scripts

Robin Zhang and Labs 2025

Expressions and Scripts

In our creative workflow, DigiShow mainly acts as a console that can connect to various media control signals, and completes the production of performance programs with other creative software through signal mapping. In this process, code programming is generally not required, which is an important feature of DigiShow. However, DigiShow also provides more possibilities for users who are accustomed to code programming.

DigiShow 1.5 and later versions support JavaScript / Qml-based expressions and scripting environments. Some simple JavaScript programming can greatly enhance the ability to implement interactive logic in DigiShow, and can also simplify some functions that originally required collaboration with external software to be completed using only DigiShow.



This section is intended for

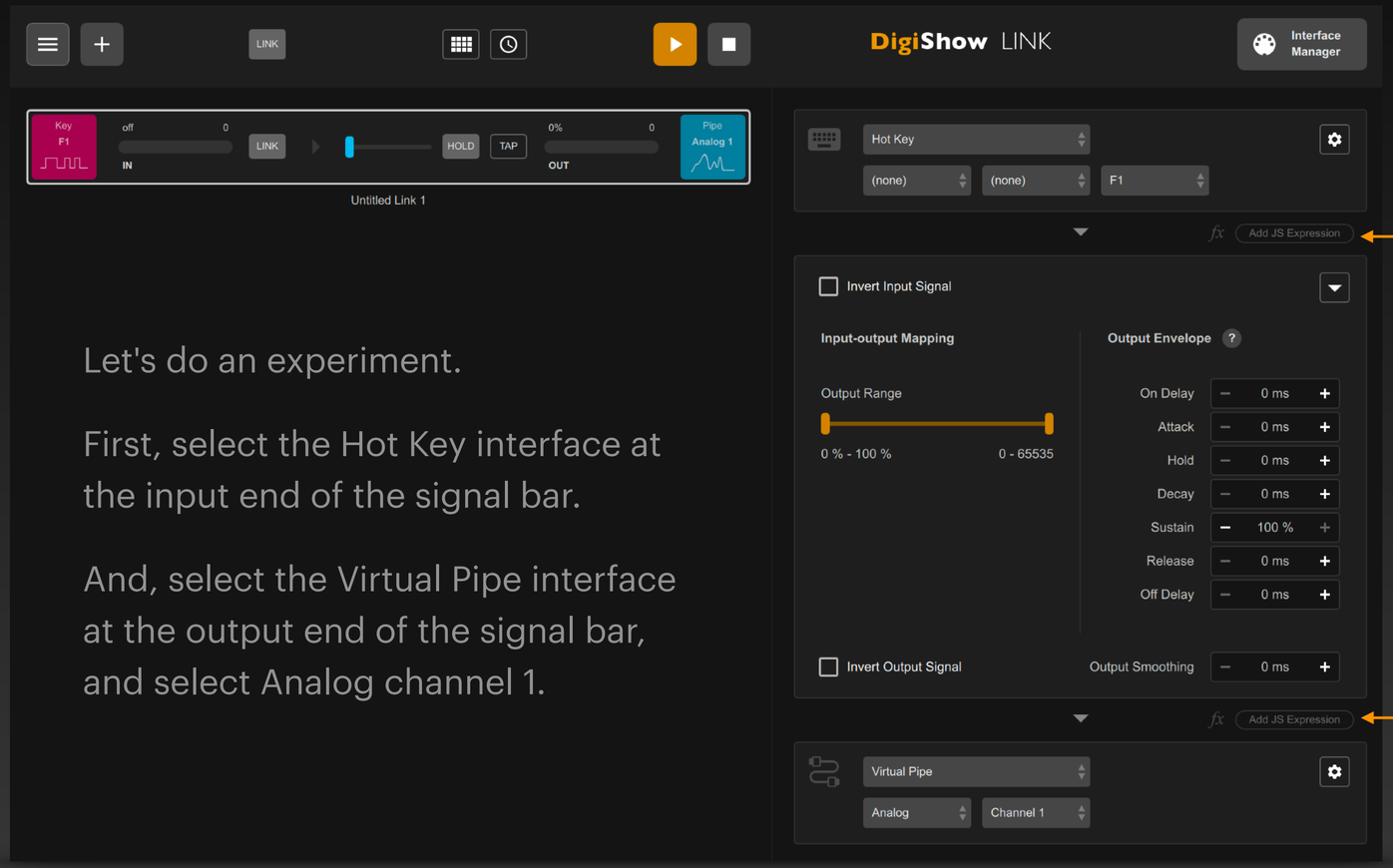
people with some JavaScript programming skills,

but others can also gain a general understanding.

Signal Input and Output Expressions

Adding Expressions

You can add a JS expression to any signal bar input or output:



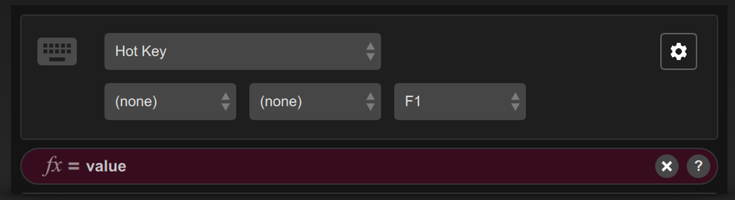
Let's do an experiment.

First, select the Hot Key interface at the input end of the signal bar.

And, select the Virtual Pipe interface at the output end of the signal bar, and select Analog channel 1.

1

Click this button to add a JS expression for the signal input.



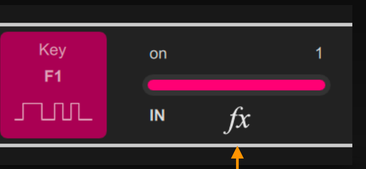
The *fx* text input box appears for entering an expression

2

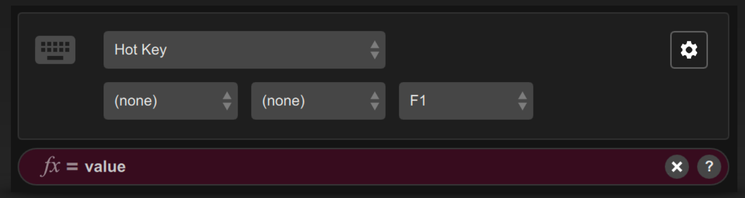
Click this button to add a JS expression for the signal output

Writing Expressions

In the *fx* text input box, enter an expression, which is a calculation formula that can change the value of the signal. The expression can contain some specific variables, numbers, mathematical operators and functions, etc. The expression needs to conform to the syntax of the JavaScript language.

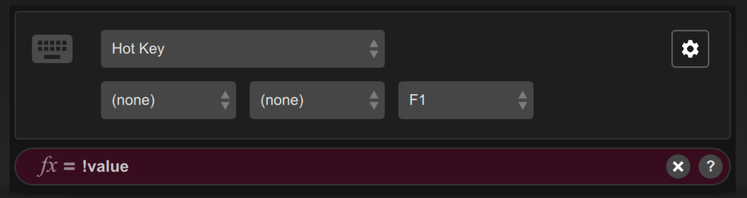


After adding the expression, a small *fx* icon appears in the signal value display



1. The default expression in the text input box is value. Here, value is a variable name, which refers to the original value of this signal.

   In this example, the signal state is 1 when the Hot Key is pressed, and the signal state is 0 when the key is released.

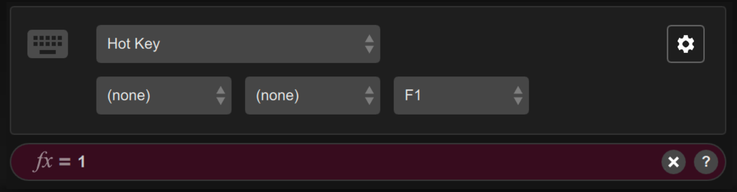


2. Changing the expression to !value will invert the value of the signal. In this example, the signal state is 0 when the Hot Key is pressed and 1 when the Hot Key is released.

   Press Enter to confirm the expression.

The expression is evaluated when the signal input changes. When the expression fails to evaluate, a warning sign appears and the signal retains its original value.

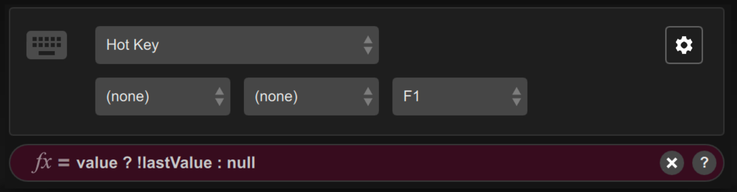
Writing Expressions



3

Changing the expression to 1 means that the value of this signal is always 1. In this example, once the Hot Key is pressed, the signal state will be locked at 1.

The variable lastValue refers to the signal value before the input changes, and !lastValue means reversing the last signal state value.



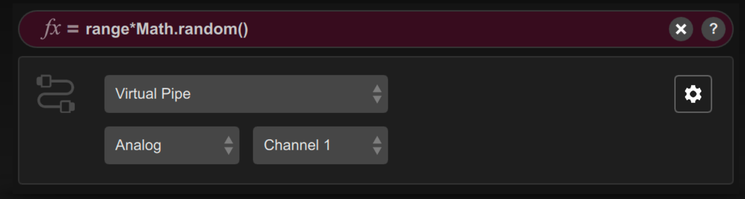
4

Change the expression to value ? !lastValue : null.
In this example, it works as follows: press the button once and the signal state changes to 1, press the button again and the signal state changes to 0, and continue to press the button to reverse the signal state again.

In this expression, the ? : operator is used to judge the condition and return different values. Its syntax is: <condition> ? <value 1> : <value 2>
The value 1 will be returned if the condition evaluates to true, and the value 2 will be returned if the condition evaluates to false.

In this example, this expression logic is: when the button is pressed, value is equal to 1 (true), and !lastValue is returned; when the button is released, value is equal to 0 (false), and null is returned, that is, the signal state value does not change.

Writing Expressions

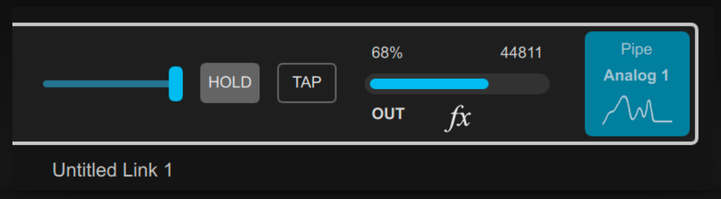


5 Add the expression at the output of the signal bar: range*Math.random()

The variable range refers to the value range of this signal. Math.random() is a function provided by the JS mathematic library that generates a random decimal between 0 and 1.

The expression can also be optimized to: value ? range*Math.random() : 0

When linked to the Hot Key signal input, the signal output generates a random number when the key is pressed, and the value returns to zero when the key is released.



6 When you move the fader in the signal bar, or when the input signal in the signal bar changes and triggers the output signal to update, the expression will perform calculations and output a random analog value ranging from 0 to 100%.

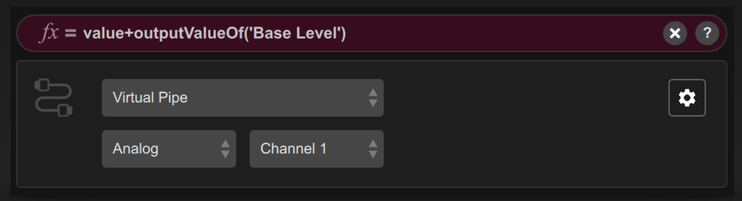
To summarize some common variables that can be included in expressions:

value refers to the original signal value before the expression is evaluated.
range refers to the maximum value of this signal in the value range.
lastValue refers to the last signal value after the expression was evaluated.
null means empty. When the expression returns null, it means that the signal value does not need to be changed.

Referencing Other Signals

Two common functions that can be used in expressions to obtain the signal values on other signal bars:

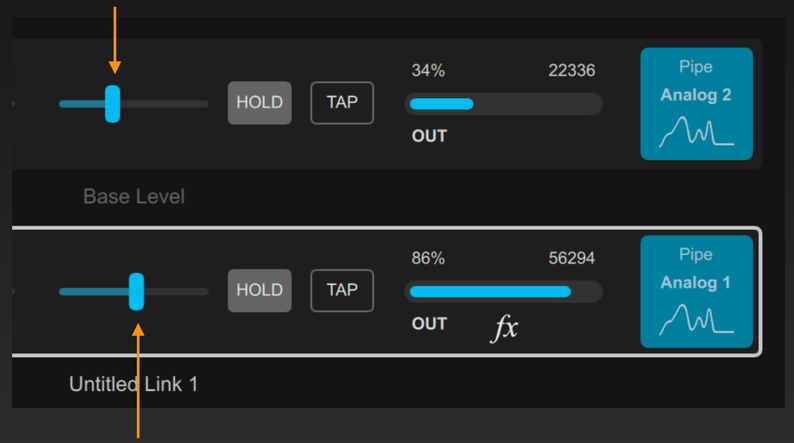
inputValueOf(name) get the input value of the signal bar with title name
outputValueOf(name) get the output value of the signal bar with title name



1 For example, write the expression at the output of the signal bar: value+outputValueOf('Base Level')

The signal output value needs to be calculated by this expression. This expression means: the set value of Base Level needs to be added to the original value of the signal before outputting it.

2 Add an extra signal bar in the signal link table, select the Virtual Pipe interface at its output, and select Analog channel 2. Then change the title of the signal bar to 'Base Level'. Now move the fader to set the Base Level value.



3 When you move the fader in the signal bar, or when the input signal in the signal bar changes and triggers the output signal to update, the expression will perform the calculation and output the calculated analog value.

Displaying Information Prompt

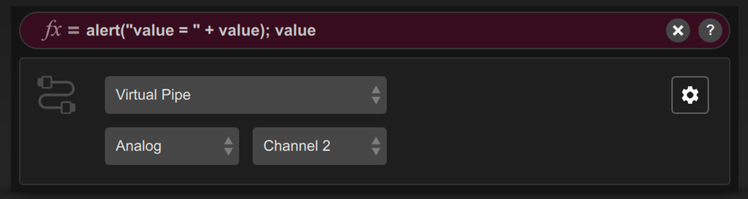
Two functions that can be used in expressions to show information in the window:

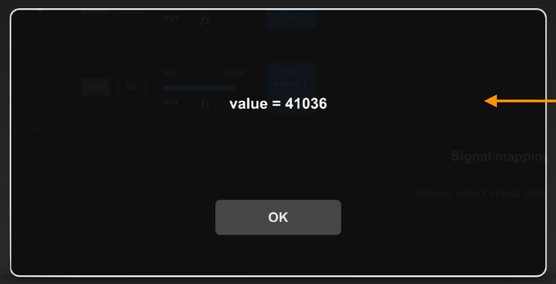
alert(message) show a message dialog box with the text message
toast(message) pop up a notification prompt box with the text message

alert() 和 toast() 也常用于在表达式或脚本中进行代码调试时的信息提示。

An expression can contain multiple statements, which need to be separated by semicolons. The last statement is used to return the value calculated by the expression. For example:

alert("value = " + value); value





1

In this example, an alert() message box is displayed when the signal changes.

And closes it after pressing the OK button.

2

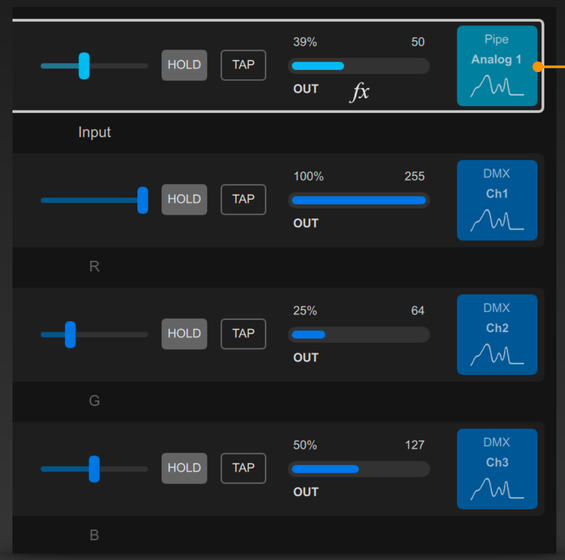
In this example, a toast() prompt box is displayed when the signal changes.

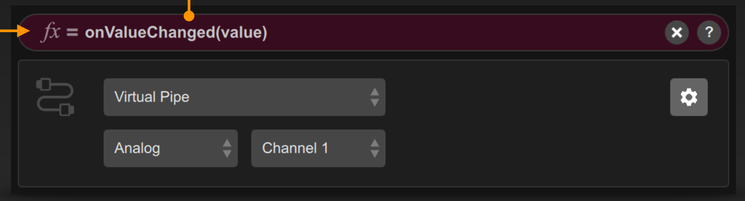
The box will disappear after a few seconds.

User-defined Functions and Scripting Environment

User-defined Functions

When you need to implement more complex or repetitive logic in expressions, you usually have to prepare your own functions in advance and then call them in expressions. At this time, you need to create a script file for your project so that you can write the program code of the user-defined functions in it.






```
MyShow.scriptable.txt
DigishowScriptable {

    function onValueChanged(value) {
        if (value < 32) {
            setOutputValueOf('R', 127)
            setOutputValueOf('G', 255)
            setOutputValueOf('B', 64)
        } else if (value < 64) {
            setOutputValueOf('R', 255)
            setOutputValueOf('G', 64)
            setOutputValueOf('B', 127)
        } else if (value < 96) {
            setOutputValueOf('R', 64)
            setOutputValueOf('G', 127)
            setOutputValueOf('B', 255)
        } else {
            setOutputValueOf('R', 0)
            setOutputValueOf('G', 0)
            setOutputValueOf('B', 0)
        }
        return value
    }
```

When the signal changes at this signal bar output, executes the expression: onValueChange(value)

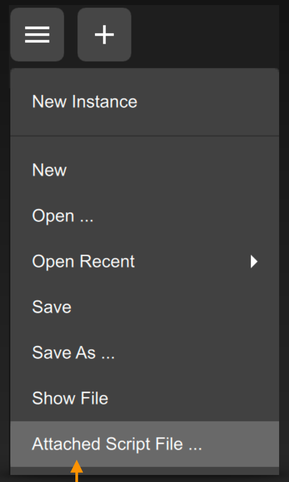
In this example, the definition of the onValueChange() function is saved in an attached script file bundled with the current project.

A common function that can be used in scripts:

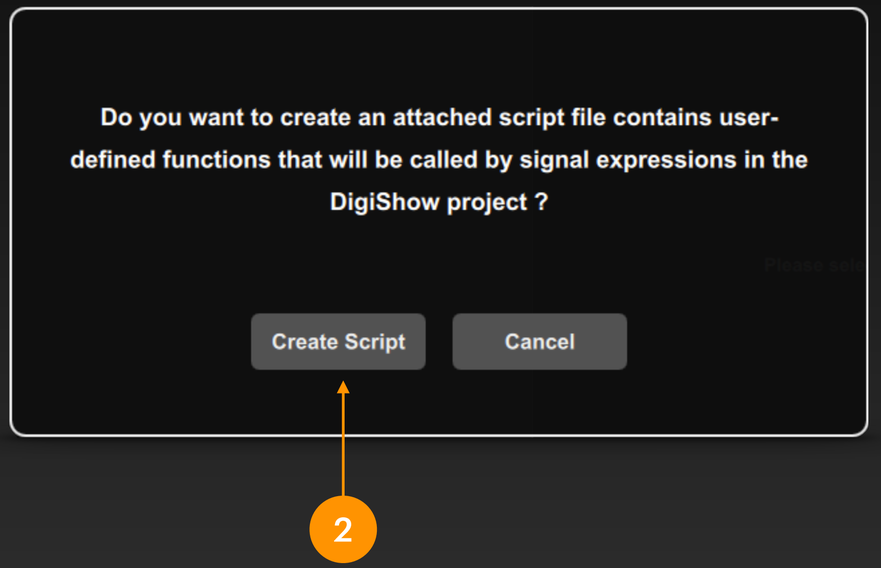
setOutputValueOf(name, value)
Change the output value of the signal bar with title name to value.

Creating Script Files

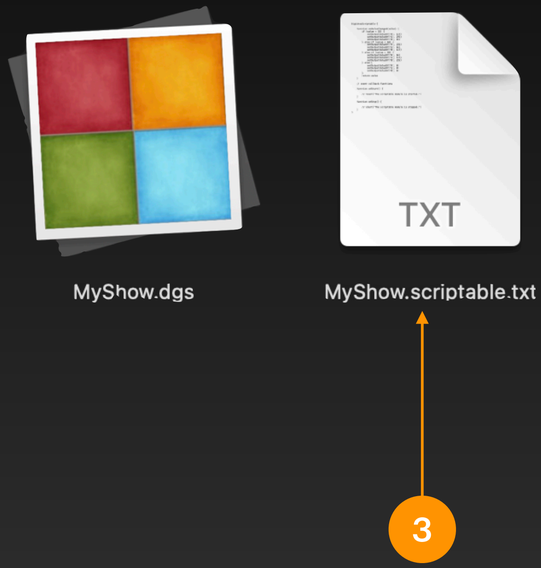
After saving a project in DigiShow, you can create an attached script file for this project.



Click the
Attached Script Files...
item in the menu.



And click the
Create Script button in
the pop-up dialog box



A script file with the same name as the current project and a .scriptable.txt suffix will be generated in the directory where the current project file is located. You can open it with a text editor or a code editor (such as VSCode). The file format complies with the qml code specification.

Using Script Files

As long as the DigiShow project is started, the attached script file is also loaded. All functions and properties (variables) defined in the script can be applied in the signal expressions.

The script file generally contains two special functions: onStart() is called by the system when the project starts, and onStop() is called by the system when the project stops. You can modify the code implementation according to your needs.

```
DigishowScriptable {
    function onStart() {
        toast('The scriptable module is started.')
    }
    function onStop() {
        alert('The scriptable module is stopped.')
    }
}
```

DigiShow script files use the Qml (Qt Modeling Language) scripting language specification.

In the script, you can use JavaScript to write your own functions in the DigishowScriptable { ... } code segment, and you can also use Qml syntax to declare properties (variables). In this scripting environment, you can also call more abundant DigiShow functions, such as:

```
app.slotTitled('test').setSlotOption('outputSmoothing', 1000)
```

That is, set the signal mapping parameter Output Smoothing in the signal bar titled 'test' to 1000 milliseconds

For more information about programming in expressions and script environments, refer to: https://github.com/robinz-labs/digishow/blob/master/guides/expression.md

Summary

- Learn to use expressions at the input and output of the signal bar to implement more interactive logic
- Learn to write user-defined functions in attached script files for calling in expressions