

MXwendler

Javascript Interface Description Version 2.0

This document describes the MXWendler (MXW) Javascript Command Interface. You will learn how to control MXWendler through the Javascript interface. We will describe parts of the hierarchical control approach and how to create interactive Javascripts in MXWendler.

11.07, valid for builds > 1645

Contents

What is Javascript?.....	1
Why should i control MXWendler using Javascript?.....	1
Where do i enter Javascript Scripts?.....	1
Which interface elements can i control using Javascript?.....	2
Example Script 1: Print Frame Number, Make Screenshot, Start Clip.....	4
Example Script 2: Reading/Setting SWF Flash Media Variables.....	5

What is Javascript?

Javascript, also known as ECMA Script is a standardized interpreted programming language. It is mainly used in webbrowsers to control web pages and test user input before sending.

Why should i control MXWendler using Javascript?

Although there are many ways to control MXWendler with a lot of I/O devices, beginning with the mouse, the keyboard, DMX dongles and MIDI devices, and there is a broad automation layer in MXWendler using events and playlists, **some desired behaviour is simply too complex** to be offered through a GUI element. Javascript allows to program any animation and behaviour sequence. Additionally, Javascript allows to **emit values to I/O devices like DMX or MIDI** (please check your version for this functionality) and set various internal states in media, eg. **set variables and exchange text strings in SWF files**.

Where do i enter Javascript Scripts?

Scripts reside always close to the events. Scripts have three entry points:

```
void on_trigger ( float triggervalue )
{
    //
    // this function will be called
    // once after an event occurred
    // and before drawing the frame
    // triggervalue is the normalized
    // value that the mapped control
    // device emits:
    // keyboard ( press/release )    [1:0]
    // midi 0 .. 127                 [0 .. 1]
```

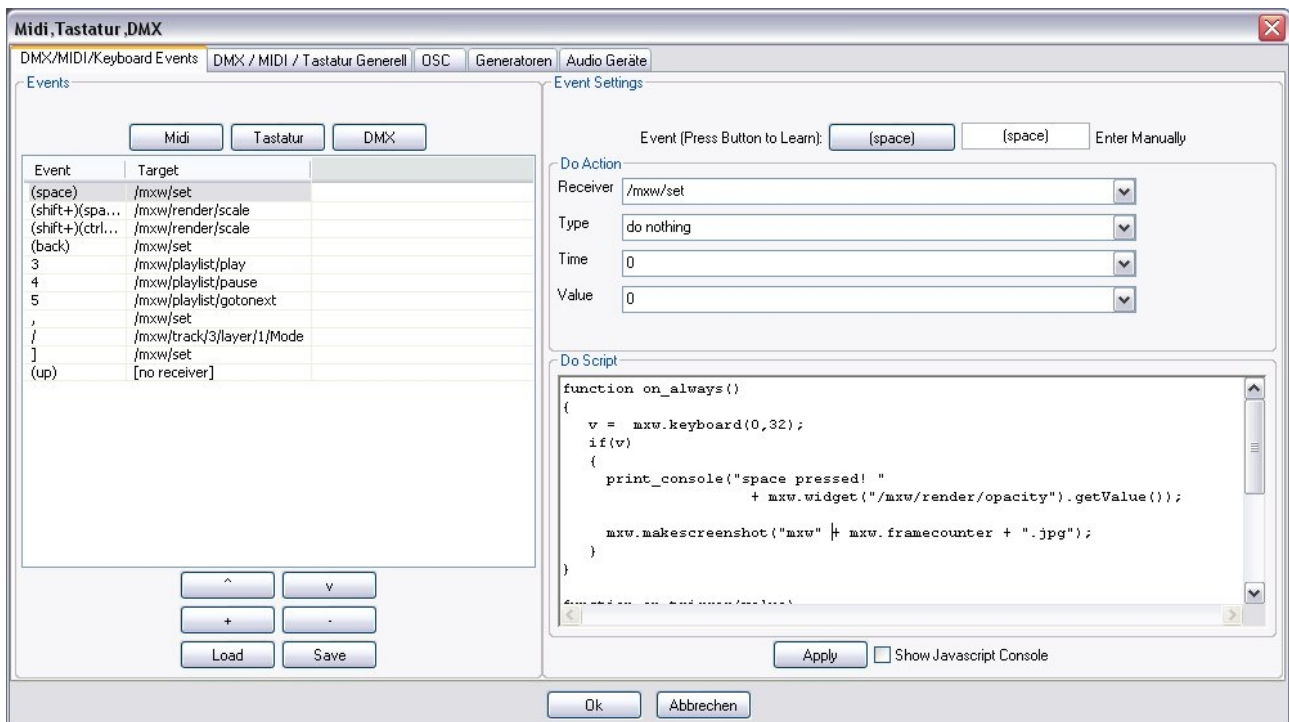
```

        // dmx 0 .. 255                                [0 .. 1]
    }

float on_repeat(void)
{
    //
    // this function will be called
    // the next frame after on_trigger
    // has been called.
    // this function will be called
    // repeatedly as long as the function
    // returns a positive value
}

void on_always(void)
{
    //
    // this function will be called
    // once each frame regardless
    // of any inner or outer state.
    // be careful with this function,
    // the only way to turn it off
    // is to remove this script and
    // reload the event map
}

```



Which interface elements can i control using Javascript?

You can control any element you can adress using the standard hierarchical MXWendler access pattern. To access MXWendler elements, you access the static object "mxw" - it is an object like

the javascript Math object, which is always available. To facilitate debugging, a script-wide function "print_console" is available, too.

```
function on_tigger(value)
{
    print_console("on_trigger called with the value " + val);
}
```

The object "mxw" offers the following members and properties:

```
function on_tigger(value)
{

    //
    // print various states
    print_console("current system time "
        + mxw.millis
    );
    print_console("current system frame width "
        + mxw.width
    );
    print_console("current system frame height "
        + mxw.height
    );
    print_console("current system output width "
        + mxw.outwidth
    );
    print_console("current system output height "
        + mxw.outheight
    );
    print_console("current system framecounter "
        + mxw.framecounter
    );

    //
    // read-access to IO devices
    // sending IO values is currently not supported ( 11/07 )
    print_console("current dmx value at channel 413 "
        + mxw.dmx(413)
    );
    print_console("current midi value device 1 channel 10 "
        + mxw.midi(1,10)
    );
    print_console("current keyboard value for (shift)+(space) "
        + mxw.keyboard(1,32)
    );

    //
    // make a screenshot
    print_console("make a screenshot named screen.jpg "
        + mxw.screenshot("screen.jpg")
    );
}
```

```

//
// access widgets and control and get/set
// values
print_console("access opacity control of main render output "
              + mxw.widget("/mxw/render/opacity")
              );

print_console("main render opacity value is "
              + mxw.widget("/mxw/render/opacity").getValue()
              );

print_console("main render opacity: set 0.5 value");
mxw.widget("/mxw/render/opacity").setValue(0.5);

}

```

Example Script 1: Print Frame Number, Make Screenshot, Start Clip

This script will:

- always print the current frame number
- on trigger, create a screenshot
- 2 seconds after triggering load the set with number 10

```

var load_set_at;

function on_always()
{
    //
    // read current frame and print to console
    print_console("current frame: " + mxw.framecounter );
}

function on_trigger(value)
{
    //
    // create screenshot with name
    // like "mxw_20125.jpg"
    makescreenshot("mxw_" + mxw.millis + ".jpg");

    //
    // store the triggering point in time,
    // add time distance ( 2000 msec = 2 seconds )
    load_set_at = mxw.millis + 2000;
}

function on_repeat()
{
    //
    // this function will be called
    // each frame after on_trigger as
    // log as it returns a value higher
    // than 0
    // we calc the time distance and

```

```

// use the result to trigger
// loading a set

ret = load_set_at - mxw.millis;

if(ret>0)
{
    //
    // we are early, return value is positive,
    // we will be called again
    return ret;
}
else
{
    //
    // ret is lower 0, more than
    // two seconds since trigger
    // so start action

    //
    // let js resolve widget, then
    // send value as command
    mxw.widget("/mxw/set").setValue(10);

    //
    // return 0: we are done
    return 0;
}
}

```

Example Script 2: Reading/Setting SWF Flash Media Variables

This script will:

- check for a certain clip
- check for its media
- set a variable
- read that variable and print it to the console

```

//
// only on_trigger is needed
function on_trigger(value)
{
    //
    // get reference of clip
    // *never* store this value, it is
    // highly dynamic and may change each frame!
    w = mxw.widget("/mxw/track/active/layer/active/clip");

    //
    // always check for existence
    if(w)
    {

```

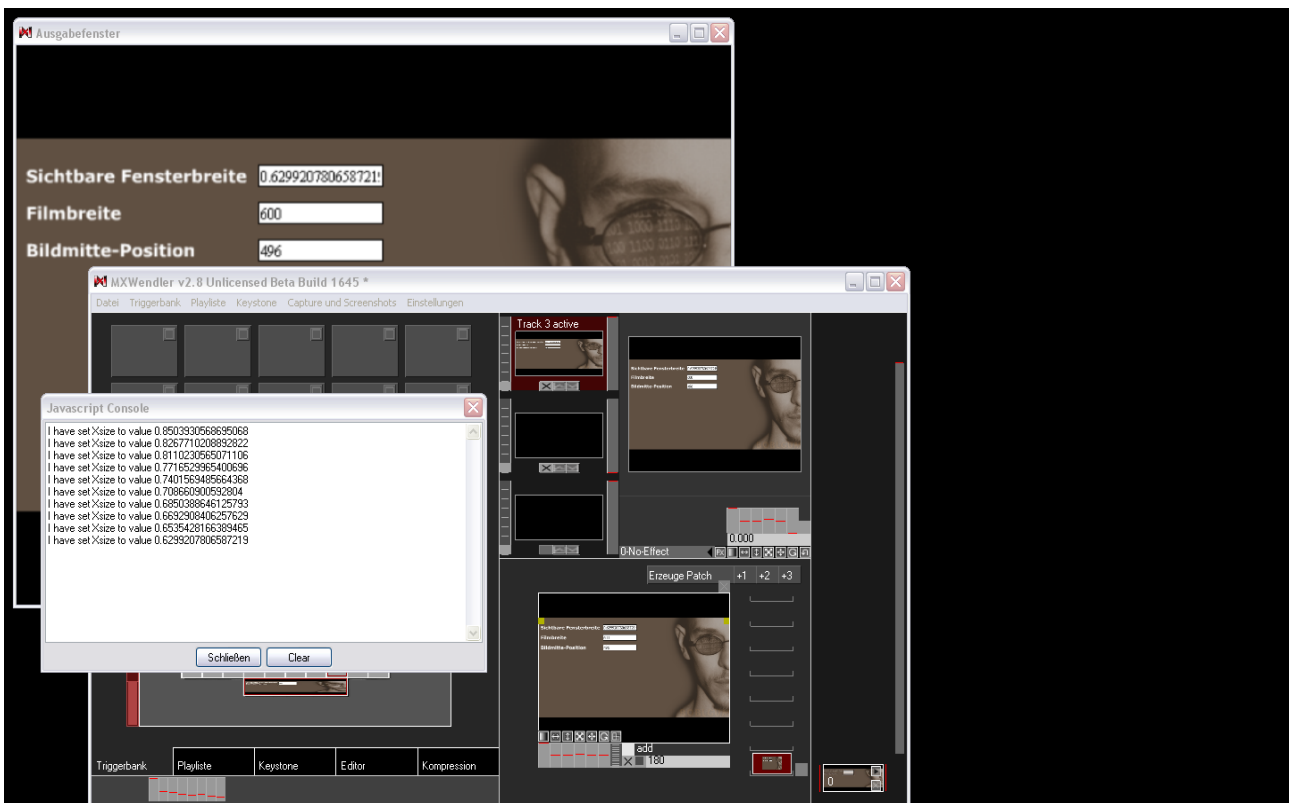
```

//
// get media ref. *never* store this value, too.
m = w.media();
if(m)
{
    //
    // the name of the flash variable.
    // you have to set this name in your swf
    // creation tool.
    var swfvar = "Xsize";

    //
    // set the value, always passed as string.
    // you can set numeric values here or pass
    // names, titles, etc.
    m.setVariable(swfvar,value);

    //
    // get the value, always returned as string
    print_console( "I have set "+ swfvar + " to value " +
        m.getVariable(swfvar)
    );
}
}
}

```



Setting SWF Variables through the Javascript Interface