BrightSign®

TECHNICAL NOTES

Using JavaScript Objects for BrightScript (FW 6.0.x)

BrightSign, LLC. 16780 Lark Ave., Suite B Los Gatos, CA 95032
408-852-9263 | www.brightsign.biz

TABLE OF CONTENTS

INTRODUCTION

The BrightSign HTML engine exposes several JavaScript objects for BrightScript. These objects allow you to link many standard interactive, database, and hardware elements (serial, CEC, device info, etc.) to HTML pages. This tech note details the functions and parameters for each JavaScript object. For more information about the BrightScript objects that the JavaScript objects are linked to, see the BrightScript Object Reference Manual.

Enabling BrightScript JavaScript Objects

For security reasons, all BrightScript JavaScript objects are disabled by default. As a result, you will encounter DOM errors like those shown below if you do not enable first enable them (to view a console log of BrightScript/JavaScript events, navigate to the Diagnostic Web Server or use the JavaScript console).

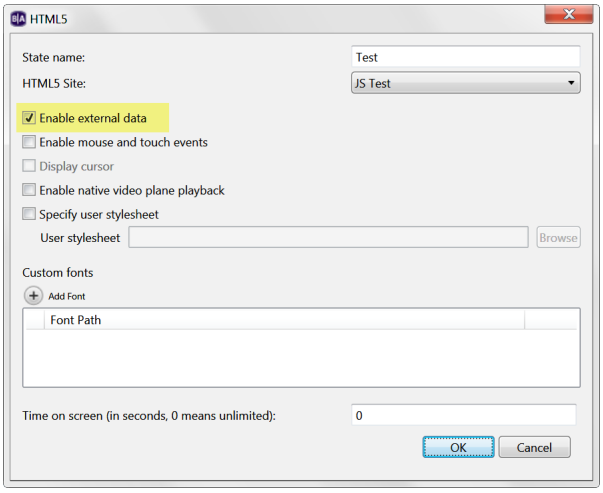
```
[  751.523] BSPLAY: file:///JS Test-/index.html
[  751.628] JS CONSOLE: file:///JS%20Test-/index.html(30): My function test start
[  751.660] JS CONSOLE: file:///JS%20Test-/index.html(31): InvalidAccessError: DOM Exception 15: A parameter or an
operation was not supported by the underlying object.
```

Follow the below steps to enable BrightScript JavaScript objects.

If Using BrightAuthor to Display HTML

Make sure that you are using BrightAuthor version 4.1 or later. Earlier versions do not have the code that enables BrightScript objects for JavaScript.

Also, ensure that the **Enable external data** box is checked within the HTML5 state that is displaying your page.



If displaying HTML using BrightScript

Call the `AllowJavaScriptUrls()` method on the *roHtmlWidget* instance you are using to display the page. This method accepts an associative array that maps JavaScript BrightScript classes to the URL(s) that are allowed to use them.

- An `all` key indicates that all classes are authorized for the associated URL(s).
- An asterisk `"*"` value indicates that all URLs are authorized for the associated BrightScript class.
- A `"local"` value indicates that all local pages are authorized for the associated BrightScript class.

Example: The following will enable all BrightScript classes for all URLs.

```
html.AllowJavaScriptUrls({ all: "*" })
```

Example: The following will enable all BrightScript classes for local pages and the BrightSign homepage.

```
html.AllowJavaScriptUrls({ all: "local", "http://www.brightsign.biz" })
```

BSControlPort

For more information about available methods, refer to the Object Reference Manual entries on *roControlPort* and *roGpioControlPort*.

Object Creation

```
BSControlPort(in DOMString PortName);
```

Methods

- `boolean SetOutputValue(in unsigned long Param)`
- `boolean SetOutputValues(in unsigned long Param1,`
  `in unsigned long Param2,`
  `in unsigned long Param3,`
  `in unsigned long Param4)`
- `boolean SetPinValue(in unsigned long Pin,`
  `in unsigned long Param)`
- `boolean ConfigureAsInput (in unsigned long Pin);`
- `boolean ConfigureAsOutput (unsigned long Pin);`

Events

The following events are available on the *BSControlPort* object. Each event can receive a `ControlPortEvent` event.

- `oncontroldown`
- `oncontrolup`
- `oncontrolevent`

ControlPortEvent – Attributes

- `readonly attribute unsigned long code`

Example

The following JavaScript example causes the LEDs on a BP900 button board to twinkle:

```
var bp900_gpio;
function myFunction()
{
    var bp900_setup = new BSControlPort("TouchBoard-0-LED-SETUP");
    bp900_setup.SetPinValue(0, 11)

    var bp900 = new BSControlPort("TouchBoard-0-LED");
    bp900.SetPinValue(0, 0x07fe)
    bp900.SetPinValue(1, 0x07fd)
    bp900.SetPinValue(2, 0x07fb)
    bp900.SetPinValue(3, 0x07f7)
    bp900.SetPinValue(4, 0x07ef)
    bp900.SetPinValue(5, 0x07df)
```

```
    bp900.SetPinValue(6, 0x07bf)
    bp900.SetPinValue(7, 0x077f)
    bp900.SetPinValue(8, 0x06ff)
    bp900.SetPinValue(9, 0x05ff)
    bp900.SetPinValue(10, 0x03ff)

    bp900_gpio = new BSControlPort("TouchBoard-0-GPIO");
    bp900_gpio.oncontroldown = function(e)
    {
        console.log('###### oncontroldown' + e.code);
    }
}
```

Example

The following example displays events from a GPIO expander board. Note that using the equivalent *roGpioControlPort* object in BrightScript at the same time will result in unpredictable I/O behavior.

```
<html>
    <head>
    <script>
        var bp900_gpio = new BSControlPort("BrightSign");

        bp900_gpio.oncontroldown = function(e){
            console.log('### oncontroldown ' + e.code);

            newtext = " DOWN: " + e.code + "\n";
            document.myform.outputtext.value += newtext;
            document.myform.outputtext.scrollTop =
document.myform.outputtext.scrollHeight;
        }

        bp900_gpio.oncontrolup = function(e){
            console.log('### oncontrolup ' + e.code);

            newtext = "   UP: " + e.code + "\n";
            document.myform.outputtext.value += newtext;
            document.myform.outputtext.scrollTop =
document.myform.outputtext.scrollHeight;
        }

        bp900_gpio.oncontrolevent = function(e){
            console.log('### oncontrolevent ' + e.code);

            newtext = "EVENT: " + e.code + "\n";
            document.myform.outputtext.value += newtext;
            document.myform.outputtext.scrollTop =
document.myform.outputtext.scrollHeight;
        }

        function ledOn(output_index)
        {
            bp900_gpio.ConfigureAsOutput(output_index);
```

```
            newtext = "  LED: " + output_index + "\n";
            document.myform.outputtext.value += newtext;
            document.myform.outputtext.scrollTop =
document.myform.outputtext.scrollHeight;
        }

        function buttonOn(output_index)
        {
            bp900_gpio.ConfigureAsInput(output_index);
            newtext = "INPUT: " + output_index + "\n";
            document.myform.outputtext.value += newtext;
            document.myform.outputtext.scrollTop =
document.myform.outputtext.scrollHeight;
        }
    </script>
    </head>

    <body bgcolor="#E6E6FA">

    <h1>GPIO/Expander Test Page</h1>
    Note this does NOT play well with <em>roGpioControlPort</em>!!<br><br>
    <input type="button" onclick="ledOn(0)" value="led0">
    <input type="button" onclick="ledOn(1)" value="led1">
    <input type="button" onclick="ledOn(2)" value="led2">
    <input type="button" onclick="ledOn(3)" value="led3">
    <input type="button" onclick="ledOn(4)" value="led4">
    <input type="button" onclick="ledOn(5)" value="led5">
    <input type="button" onclick="ledOn(6)" value="led6">
    <input type="button" onclick="ledOn(7)" value="led7">
    <br>
    <input type="button" onclick="buttonOn(0)" value="button0">
    <input type="button" onclick="buttonOn(1)" value="button1">
    <input type="button" onclick="buttonOn(2)" value="button2">
    <input type="button" onclick="buttonOn(3)" value="button3">
    <input type="button" onclick="buttonOn(4)" value="button4">
    <input type="button" onclick="buttonOn(5)" value="button5">
    <input type="button" onclick="buttonOn(6)" value="button6">
    <input type="button" onclick="buttonOn(7)" value="button7">
    <br>

    <form name="myform">
        <textarea readonly rows="50" cols="100" name="outputtext" style="font-
family:monospace"></textarea>
    </form>

</body>
</html>
```

BSDatagramSocket

For more information about available methods, refer to the Object Reference Manual entry on *roDatagramReceiver* and *roDatagramEvent*.

Methods

- `boolean BindLocalPort(in unsigned long portNumber)`
- `long GetLocalPort()`
- `boolean JoinMulticastGroup(in DOMString group)`
- ```
  boolean SendTo(in DOMString dest,
        in long port,
        in ArrayBufferView data)
  ```
- ```
  boolean SendTo(in DOMString dest,
        in long port,
        in DOMString data)
  ```

Events

The following event is available on the *BSDatagramSocket* object. It can receive an event of the type `DatagramSocketEvent`. Use `GetBytes()` to retrieve the body of the UDP message.

- `ondatagram`

DatagramSocketEvent

The DatagramSocketEvent has the following attributes:

- `readonly attribute DOMString remoteHost`
- `readonly attribute int remotePort`

The DatagramSocketEvent supports the following methods:

- `ArrayBuffer getBytes()`

Example

```
var bsSocketMessage = new BSDatagramSocket();
bsSocketMessage.BindLocalPort(1234)
bsSocketMessage.ondatagram = function(e){
   console.log(e);
};
```

BSDeviceInfo

For more information about available methods, refer to the Object Reference Manual entry on *roDeviceInfo*.

Attributes

- `readonly attribute DOMString model`
- `readonly attribute DOMString version`
- `readonly attribute int deviceUptime`
- `readonly attribute int deviceLifetime`
- `readonly attribute int deviceBootCount`
- `readonly attribute DOMString bootVersion`
- `readonly attribute DOMString deviceUniqueId`
- `readonly attribute DOMString family`

Methods

- `int VersionCompare(in DOMString version)`
- `int BootVersionCompare(in DOMString version)`
- `boolean HasFeature(in DOMString feature)`

Example

The following JavaScript example posts device information on the page when the button is clicked:

```
function deviceInfo()
{
    var device_info = new BSDeviceInfo();

    document.getElementById("modelText").innerHTML = device_info.model;
    document.getElementById("versionText").innerHTML = device_info.version;
    document.getElementById("bversionText").innerHTML = device_info.bootVersion;
    document.getElementById("serialText").innerHTML = device_info.deviceUniqueId;
    document.getElementById("familyText").innerHTML = device_info.family;

    document.getElementById("uptime").innerHTML = device_info.deviceUptime;
    document.getElementById("lifetime").innerHTML = device_info.deviceLifetime;
    document.getElementById("bootcount").innerHTML = device_info.deviceBootCount;

    if(device_info.VersionCompare("4.7.36") > 0)
    {
        document.getElementById("version1").innerHTML = "Version > 4.7.36"
    }
    else
    {
        document.getElementById("version1").innerHTML = "Version <= 4.7.36"
    }

    if(device_info.HasFeature("Six Channel Audio"))
```

```
    {
        document.getElementById("feature").innerHTML = "6 Channel Audio
Available"
    }
    else
    {
        document.getElementById("feature").innerHTML = "6 Channel Audio NOT
Available"
    }
}
```

BSIRReceiver

This class receives IR events. For more information, refer to the Object Reference Manual entry on *roIRReceiver*.

Object Creation

A *BSIRReceiver* object must specify the hardware interface that will receive IR events, as well as the encoding(s) to look for:

```
BSIRReceiver(in DOMString interface, in DOMString encodings);
```

Valid hardware interfaces include the following:

- `"GPIO"`: Pin 1 of the GPIO connector
- `"IR-in"`: The 3.5mm IR input/output connector (available on the 4K series of players)
- `"Iguana"`: The Iguanaworks IR transceiver. This source can support both NEC and RC5 encodings simultaneously.

Valid encodings include the following (multiple encodings can be specified in the string using a ";"):

- `"NEC"`
- `"RC5"` (supported on the Iguanaworks device only)

Events

These events are available on the *BSIRReceiver* object. Each event can receive an `IRReceiverEvent` event.

- `onremotedown`
- `onremoterepeat`
- `onremoteup` (supported on the Iguanaworks device only)

IRReceiverEvent – Attributes

- `readonly attribute DOMString irType;`
- `readonly attribute unsigned long code;`

Example

The following JavaScript example displays messages on the log when receiving remote codes:

```
function myFunction()
{
    var ir_receiver = new BSIRReceiver();

    ir_receiver.onremotedown = function(e){
        console.log('############ onremotedown: ' + e.irType + " - " + e.code);
    }

    ir_receiver.onremoteup = function(e){
        console.log('############ onremoteup: ' + e.irType + " - " + e.code);
    }
```

```
}
```

BSIRTransmitter

For more information about available methods, refer to the Object Reference Manual entry on *roIRTransmitter*.

Object Creation

A *BSIRTransmitter* object must specify the hardware interface that will output the IR signal:

```
BSIRTransmitter (in DOMString interface);
```

Valid hardware interfaces include the following:

- `"IR-out"`: The 3.5mm IR input connector (available on XD players) or 3.5mm IR input/output connector (available on 4K players)
- `"Iguana"`: The Iguanaworks IR transceiver

Methods

- `boolean Send(in DOMString Type, in unsigned long code)`
- `boolean SetSendPolarity(in boolean polarity)`

Example

The following JavaScript example sends the indicated IR codes when the corresponding functions are called:

```
<script>
    var irTransmitter = new BSIRTransmitter();

    function irCode1()
    {
        console.log('############ irCode1');
        irTransmitter.Send("NEC", 65284);
    }

    function irCode2()
    {
        console.log('############ irCode2');
        irTransmitter.Send("NEC", 65288);
    }

    function irCode3()
    {
        console.log('############ irCode3');
        irTransmitter.Send("NEC", 65290);
    }
</script>
```

BSVideoMode

For more information about available methods, please refer to the Object Reference Manual entry on *roVideoMode*. If you'd like to change the video mode of the player, you will need to use BrightScript instead of this JavaScript class.

Attributes

- `readonly attribute int resX`
- `readonly attribute int resY`
- `readonly attribute int safeX`
- `readonly attribute int safeY`
- `readonly attribute int safeWidth`
- `readonly attribute int safeHeight`
- `readonly attribute DOMString mode`

Methods

- `boolean IsAttached(in DOMString connector)`
- `DOMString GetBestMode(in DOMString connector)`
- `boolean SetBackgroundColour(in unsigned long rgb)`
- `boolean SetBackgroundColour(in unsigned long r,`
  `in unsigned long g,`
  `in unsigned long b)`
- `boolean HdmiAudioDisable(in boolean disable)`

Example

The following JavaScript example illustrates how to retrieve information about the current video mode:

```
    function fillInVideoData()
    {
        var videomode_info = new BSVideoMode();
        document.getElementById("resX").innerHTML = videomode_info.resX;
        document.getElementById("resY").innerHTML = videomode_info.resY;
        document.getElementById("safeX").innerHTML = videomode_info.safeX;
        document.getElementById("safeY").innerHTML = videomode_info.safeY;
        document.getElementById("safeWidth").innerHTML =
videomode_info.safeWidth;
        document.getElementById("safeHeight").innerHTML =
videomode_info.safeHeight;
        document.getElementById("videoMode").innerHTML = videomode_info.mode;
        document.getElementById("bestMode").innerHTML =
videomode_info.GetBestMode("hdmi");

        document.getElementById("connectedFlag").innerHTML =
videomode_info.IsAttached("vga");
    }
```

```
function changeBackground()
{
    var videomode_info = new BSVideoMode();
    videomode_info.SetBackgroundColour(0xFF0000);
}
```

BSCECTransmitter

For more information about available methods, refer to the Object Reference Manual entry on *roCecInterface*. Note that you can only use this JavaScript class to send CEC messages.

Methods

- `boolean SendRawMessage(in ArrayBuffer data)`
- `boolean SendRawMessage(in ArrayBufferView data)`
- `boolean SendRawMessage(in DOMString data)`

Example

The following JavaScript example shows how to send a set of CEC messages:

```
function cecDisplayOn()
{
    console.log("### cecDisplayOn ###");

    var cec_control = new BSCECTransmitter();

    var buffer = new Uint8Array(2);
    buffer[ 0 ] = 0x40;
    buffer[ 1 ] = 0x0D;

    cec_control.SendRawMessage(buffer);
}
function cecDisplayOff()
{
    console.log("### cecDisplayOff ###");

    var cec_control = new BSCECTransmitter();

    var buffer = new Uint8Array(2);
    buffer[ 0 ] = 0x40;
    buffer[ 1 ] = 0x36;

    cec_control.SendRawMessage(buffer);
}
```

BSCECReceiver

For more information about available methods, refer to the Object Reference Manual entry for *roCecInterface*. Note that you can only use this JavaScript class to receive CEC messages.

Events

The following events are available on the *BSCECReceiver* object. They can receive an event of the type *CECReceiverEvent*.

- `onececrxframe`
- `oncectxcomplete`

CECReceiverEvent – Attributes

- `readonly attribute int status`
- `readonly attribute DOMString data`

Example

```
<script>
    function loadCec()
    {
        console.log("*** loadCec ***");

        var cec_in = new BSCECReceiver();

        cec_in.oncecrxframe = function(e){
           console.log('############ oncecrxframe: ' + e.data);
        }

        serial_out.oncectxcomplete = function(e){
           console.log('############ oncectxcomplete: ' + e.status);
        }
    }
</script>
```

BSSyncManager

For more information about available methods, refer to the Object Reference Manual entry for *roSyncManager*.

Object Creation

```
BSSyncManager(in DOMString domain, in DOMString multicast_address, in
DOMString multicast_port);
```

Methods

- `BSSyncManager(in DOMString domain, in DOMString multicast_address, in DOMString multicast_port)`
- `void SetMaster(in boolean master_mode) raises(DOMException)`
- `void Synchronize(in DOMString id, in unsigned long ms_delay) raises(DOMException)`

Events

The following event is available on the *BSSyncManager* object. It can receive events of the type *BSSyncManagerEvent*.

- `onsyncevent`

HTML video tags have been extended to include a `setSyncParams` function. Calling this function causes a video to synchronize with the specified sync group.

- `setSyncParams(in DOMString domain, in DOMString id, in DOMString iso_timestamp)`

BSSyncManagerEvent – Attributes

The following attributes are relevant to the `onsyncevent` event:

- `readonly attribute DOMString domain`
- `readonly attribute DOMString id`
- `readonly attribute DOMString iso_timestamp`

Example

The following JavaScript example contains two videos being synchronized locally with *BSSyncManager*. If a Slave player is configured to be in the same PTP domain as the Master player and uses the Slave HTML script, then it will display the videos in sync with the Master player. This can be implemented on multiple Slave players.

Master Script:

```
<video id="one" hwz="on">
    <source id="one_src" src="pirates.mov">
</video>
```

```
<script>
    // Create the sync manager with provided multicast settings
    var sync = new BSSyncManager("domain1", "224.0.126.10", 1539);

    sync.onsyncevent = function (e) {

        document.getElementById("one").setSyncParams(e.domain, e.id,
e.iso_timestamp);
        document.getElementById("one").load();
        document.getElementById("one").play();
        console.log(e.domain);
        console.log(e.id);
        console.log(e.iso_timestamp);
    };

    function startTimer() {
        setTimeout(function () {
            restartMaster()
        }, 30000);
    }
    function restartMaster() {
        // Synchronize the videos to start playing in 1000ms
        sync.SetMaster(1);
        sync.Synchronize("sync_event1", 1000);
        startTimer();
    }
    restartMaster();


</script>
```

Slave Script:

```
<video id="one" hwz="on">
    <source id="one_src" src="pirates.mov">
</video>

<script>
    // Create the sync manager with provided multicast settings
    var sync = new BSSyncManager("domain1", "224.0.126.10", 1539);

    sync.onsyncevent = function (e) {

        document.getElementById("one").setSyncParams(e.domain, e.id,
e.iso_timestamp);
        document.getElementById("one").load();
        document.getElementById("one").play();
        console.log(e.domain);
        console.log(e.id);
        console.log(e.iso_timestamp);
    };
```

```
</script>

</body>
</html>
```

BSMessagePort

This object allows for an associative array to be passed from JavaScript to BrightScript (or vice versa). Only one *BSMessagePort* instance may be created per *roHtmlWidget* instance.

Methods

- `boolean PostBSMessage(in Dictionary message)`

Note: *This method does not accept nested dictionaries. The same is true for the* `PostJSMessage()` *BrightScript method.*

Events

The following event occurs when a message is sent from BrightScript to JavaScript. It will appear as a *MessagePortEvent* event.

- `onbsmessage`

Examples

The following script will send a collection of properties to BrightScript:

```
function myFunction()
{
    var bsMessage = new BSMessagePort();

    bsMessage.PostBSMessage({complete: true, result: "PASS"});
}
```

The following message will appear in BrightScript as an *roHtmlWidgetEvent*. In this case, the `GetData().reason` equals "message" and `GetData().message` contains the *roAssociativeArray*.

```
while not finished
    ev = mp.WaitMessage(30000)
    if type(ev) <> "roHtmlWidgetEvent" then
        print type(ev)
        stop
    end if

    payload = ev.GetData()
    print payload
    print "Reason: "; payload.reason
    if payload.reason = "message" then
        print "Message: "; payload.message
        if payload.message.complete = invalid then
            stop
        else if payload.message.complete = "true" then
            finished = true
            if payload.message.result = "PASS" then
                print "Test passed"
            else
                print "Test failed: "; payload.message.err
```

```
                stop
            end if
        end if
    end if
end while
```

MessagePortEvent

This event contains a single field:

```
readonly attribute any data;
```

The following script will iterate over all the fields received in the event:

```
var bsMessage = new BSMessagePort();

bsMessage.onbsmessage = function(msg)
{
    for(name in msg.data)
    {
       console.log('### ' + name + ': ' + msg.data[name]);
    }
}
```

In BrightScript, the *roHtmlWidget.PostJSMessage()* method can be used to post a message to JavaScript:

```
widget.PostJSMessage({ Param1: "Data1", Param2: "Data2", Param3: "Data3" })
```

BSSerialPort

For more information about available methods, refer to the Object Reference Manual entry on *roSerialPort*.

Object Creation

```
BSSerialPort(in unsigned long port);
```

Methods

- `void SetBaudRate(in unsigned long baudRate) raises(DOMException)`
- `void SetDataBits(in unsigned long dataBits) raises(DOMException)`
- `void SetStopBits(in unsigned long stopBits) raises(DOMException)`
- `void SetParity(in DOMString parity) raises(DOMException)`

- `boolean SetEcho(in boolean flag)`
- `boolean SetInverted(in boolean flag)`
- `boolean SetFlowControl(in boolean flag)`

- `void SetGenerateByteEvent(in boolean flag) raises(DOMException)`
- `void SetGenerateLineEvent(in boolean flag) raises(DOMException)`

- `void SetLineEnding(in DOMString eol) raises(DOMException)`

- `boolean SendByte(in unsigned long byte)`
- `boolean SendBytes(in ArrayBuffer data)`
- `boolean SendBytes(in ArrayBufferView data)`
- `boolean SendBytes(in DOMString data)`

- `boolean SendBreak(in long duration_ms)`
- `void Flush() raises(DOMException)`

Events

The following events are available via the *BSSerialPort* object. Each event can receive a `SerialPortEvent` event.

- `onserialbyte`
- `onserialline`

SerialPortEvent – Attributes

For the `onserialbyte` event:

- `readonly attribute unsigned long sbyte`

For the `onserialline` event:

- `readonly attribute DOMString sdata`

Example

```
function serialOut()
{
    console.log("*** serialOut **");

    // '2' is the first externally connected USB  port on Cheetah
    var serial_out = new BSSerialPort(2);

    serial_out.SetBaudRate(115200);
    serial_out.SetDataBits(8);
    serial_out.SetStopBits(1);
    serial_out.SetParity("none");
    serial_out.SetEcho(true);

    serial_out.SetGenerateByteEvent(true);
    serial_out.SetGenerateLineEvent(true);

    serial_out.onserialbyte = function(e){
        console.log('### onserialbyte: ' + e.byte);
    }

    serial_out.onserialline = function(e){
        console.log('### onserialline: ' + e.data);
    }

    serial_out.SendByte(89);
    serial_out.SendByte(90);
    serial_out.SendByte(91);

    serial_out.SendBytes('Hello World!');
    serial_out.SendBytes(String.fromCharCode(64, 27, 66, 67))
```

BSTicker

For more information about object creation and available methods, refer to the Object Reference Manual entry on *roTextWidget*. This object can only be used to display a "smooth right-to-left scrolling" type ticker.

Object Creation

```
BSTicker(in unsigned long x, in unsigned long y, in unsigned long w, in
unsigned long h, in long rotation);
```

The `x` and `y` integers specify the position of the ticker widget (via the top-left corner), while the `w` and `h` integers specify the size of the widget. The `rotation` parameter accepts the following values:

- `0`: 0 degrees
- `1`: 90 degrees
- `2`: 180 degrees
- `3`: 270 degrees

Once a *BSTicker* is created, you cannot change its size or positioning. Instead, destroy the *BSTicker* instance using the `ShutDown()` method and generate a new one with different parameters.

Methods

- `boolean AddString(in DOMString str);`
- `boolean PopStrings(in unsigned long count);`

- `boolean SetForegroundColor(in unsigned long argb);`
- ```
  boolean SetForegroundColor(in unsigned long a,
        in unsigned long r,
        in unsigned long g,
        in unsigned long b);
  ```

- `boolean SetBackgroundColor(in unsigned long argb);`
- ```
  boolean SetBackgroundColor(in unsigned long a,
        in unsigned long r,
        in unsigned long g,
        in unsigned long b);
  ```

- `boolean SetSeparatorString(in DOMString str);`
- `boolean SetSeparatorCircle();`
- `boolean SetSeparatorDiamond();`
- `boolean SetSeparatorSquare();`

- `unsigned long SetPixelsPerSecond(in unsigned long pps);`

- `boolean SetMultiscreen(in unsigned long offset,`
  `in unsigned long size,`
  `in DOMString ip_address,`
  `in unsigned long port);`

- `void ShutDown()`

Multiscreen Tickers

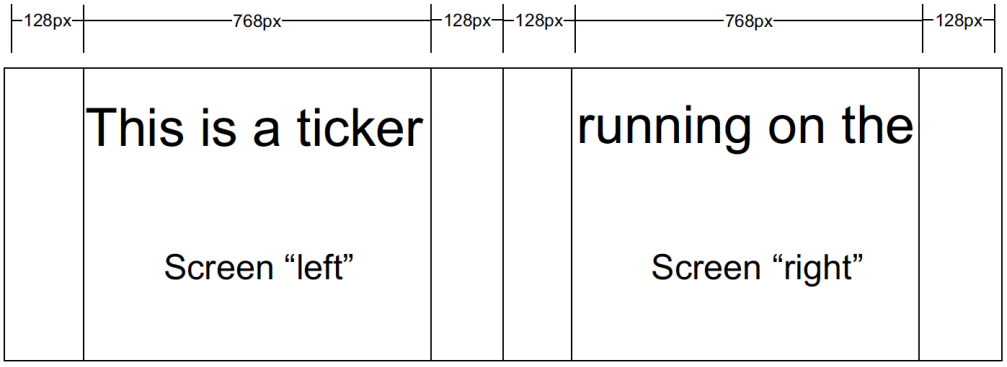
To create a ticker that scrolls across multiple screens, call the `SetMultiscreen()` method on each page.

The `size` parameter determines the total width of the multiscreen ticker across all displays. This parameter should be the same value for each `SetMultiscreen()` call. Note that the `size` of a multiscreen ticker can be different from the total width of a multiscreen display (i.e. if the width of a BSTicker widget is smaller than the width of the screen).

The `offset` parameter determines the left starting point for each ticker in the multiscreen display. The parameter value will always be 0 for the leftmost display. This parameter should be calculated using the length of the multiscreen ticker, *not the total length of the multiscreen display*.

The `ip_address` and `port` parameters refer to the multicast address that the players will use to synchronize the ticker.

The following example diagram and code specifies a multiscreen display with two monitors. The width of each monitor is 1024 pixels, while the width of each ticker is 768 pixels.




```
bsTicker = new BSTicker(128, 100,768, 100, 0);
bsTicker.SetMultiscreen(0,1536, "239.0.100.11", 5400)
var speed = bsTicker.SetPixelsPerSecond(300);
```

```
bsTicker = new BSTicker(128, 100, 768, 100, 0);
bsTicker.SetMultiscreen(768,1536,  "239.0.100.11", 5400)
var speed = bsTicker.SetPixelsPerSecond(300);
```

Example

The following script shows how to use many of the available *BSTicker* methods:

```
var bsTicker = new BSTicker(10, 110, 600, 30);
```

```
function addText1()
{
    console.log('############ addText1');
    bsTicker.AddString("addText1");
}
function addText2()
{
    console.log('############ addText2');
    bsTicker.AddString("addText2");
}
function addText3()
{
    console.log('############ addText3');
    bsTicker.AddString("addText3");
}

function setBackground1()
{
    console.log('############ setBackground1');
    bsTicker.SetBackgroundColor(0xFF0000);
}
function setBackground2()
{
    console.log('############ setBackground2');
    bsTicker.SetBackgroundColor(0x00FF00);
}

function setForeground1()
{
    console.log('############ setBackground1');
    bsTicker.SetForegroundColor(0x007700);
}
function setForeground2()
{
    console.log('############ setBackground2');
    bsTicker.SetForegroundColor(0x000077);
}

function setSeparatorString()
{
    console.log('############ setSeparatorString');
    bsTicker.SetSeparatorString(" ### ");
}
function setSeparatorCircle()
{
    console.log('############ setSeparatorCircle');
    bsTicker.SetSeparatorCircle();
}
function setSeparatorDiamond()
{
    console.log('############ setSeparatorDiamond');
```

```
    bsTicker.SetSeparatorDiamond();
}
function setSeparatorSquare()
{
    console.log('############ setSeparatorSquare');
    bsTicker.SetSeparatorSquare();
}
function setTickerSpeed()
{
    console.log('############ setTickerSpeed');
    var speed = bsTicker.SetPixelsPerSecond(180);
    console.log('############ SetPixelsPerSecond(180), Resulting speed: ' +
speed);
}
```