

# **PRIME JavaScript User Guide**

## Version 4.10.10

September 2025



Chyron PRIME JavaScript User Guide • 4.10.10 • September 2025 • This document is distributed by Chyron in online (electronic) form only, and is not available for purchase in printed form.

This document is protected under copyright law. An authorized licensee of Chyron PRIME JavaScript may reproduce this publication for the licensee's own use in learning how to use the software. This document may not be reproduced or distributed, in whole or in part, for commercial purposes, such as selling copies of this document or providing support or educational services to others.

Product specifications are subject to change without notice and this document does not represent a commitment or guarantee on the part of Chyron and associated parties. This product is subject to the terms and conditions of Chyron's software license agreement. The product may only be used in accordance with the license agreement.

Any third party software mentioned, described or referenced in this guide is the property of its respective owner. Instructions and descriptions of third party software is for informational purposes only, as related to Chyron products and does not imply ownership, authority or guarantee of any kind by Chyron and associated parties.

This document is supplied as a guide for Chyron PRIME JavaScript. Reasonable care has been taken in preparing the information it contains. However, this document may contain omissions, technical inaccuracies, or typographical errors. Chyron and associated companies do not accept responsibility of any kind for customers' losses due to the use of this document. Product specifications are subject to change without notice.

Copyright © 2025 Chyron, ChyronHego Corp. and its licensors. All rights reserved.

## Table of Contents

<b>JavaScript Introduction.....</b>	<b>5</b>
Description.....	5
ECMA Scripting Language.....	5
Java Script Effect and Resource.....	5
Execute On Load.....	6
Mode.....	6
Timeout.....	6
JavaScript Advanced Options.....	7
Enable File Loading.....	7
Enable Window Functions.....	7
Enable Host functions.....	7
Enable Extended Host functions.....	8
Java Script Parameters.....	8
<b>Keywords.....</b>	<b>10</b>
Scene.....	10
Parent.....	10
This.....	10
Project.....	10
Channel.....	10
Playout.....	10
window.....	10
host.....	10
xHost.....	10
FindObject(string name).....	10
<b>Properties.....</b>	<b>11</b>
Size.....	11
Point.....	11
Vector.....	11
Color.....	11
Font.....	12
Outline.....	12
Shadow.....	12
TimeCode.....	12
Resolution.....	13
Bindings.....	13
<b>Objects.....</b>	<b>14</b>
SceneObject.....	14
SceneObjectList.....	14

Scene.....	14
Action.....	18
Animation.....	18
AnimationList.....	19
DynamicObject.....	19
Graphic.....	19
2D Graphic.....	20
Text.....	20
Image.....	23
Clip.....	24
Group.....	25
Control.....	25
Control Panel.....	26
Button.....	26
Check Box.....	26
Text Box.....	27
Combo Box.....	27
Parameter.....	27
JavaScript.....	28
Project.....	28
Channel.....	28
ClipPlayer.....	29
Playout.....	30
Window.....	31
<b>HostFunctions.....</b>	<b>32</b>
ExtendedHostFunctions.....	32

# JavaScript Introduction

## Description

JavaScript is a robust scripting language capable of executing advanced control over multiple scopes; Scene level, Project Level, Application Level, or Master Control Panels.

## ECMA Scripting Language

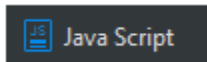
JavaScript component uses the ECMAScript Language standard as defined by the links below.

<https://developer.mozilla.org/en-US/docs/Web/JavaScript>

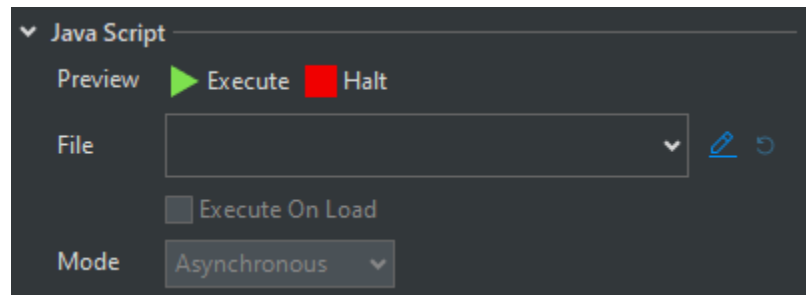
<https://tc39.es/ecma262/>

## Java Script Effect and Resource

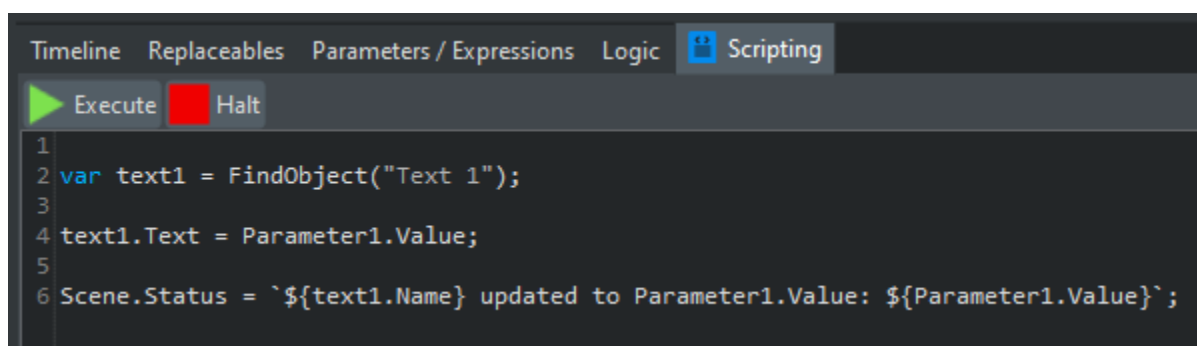
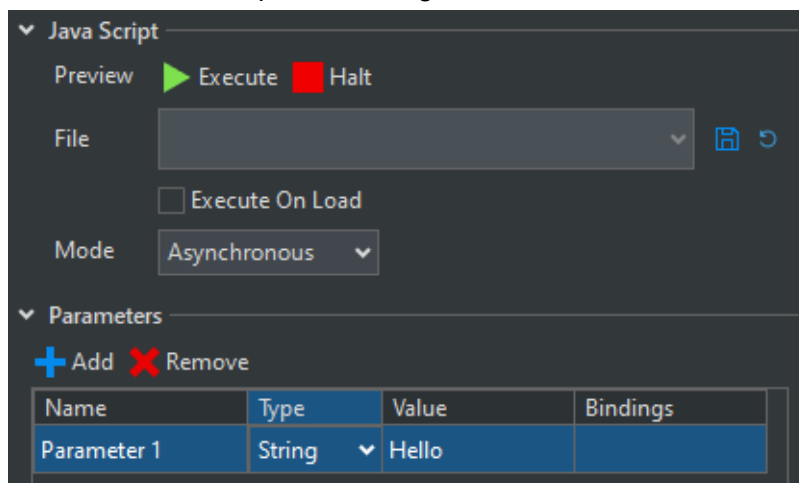
- Java Script Effect can be applied to any scene Graphic Object.
- Java Script Resource can be applied to the Scene, Project, and Application levels as well as Master Control Panels.



**To edit a JavaScript:** click the edit pencil icon. This will open the Scripting panel within Prime.

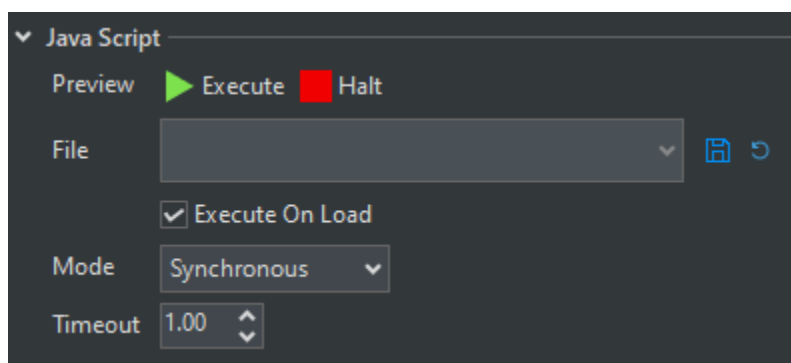


From here, you can set options, add parameters, edit the code in the Scripting window, and Execute or Halt scripts for testing.



### Execute On Load

Checking this option will cause the script to execute automatically when the scene is loaded in playout

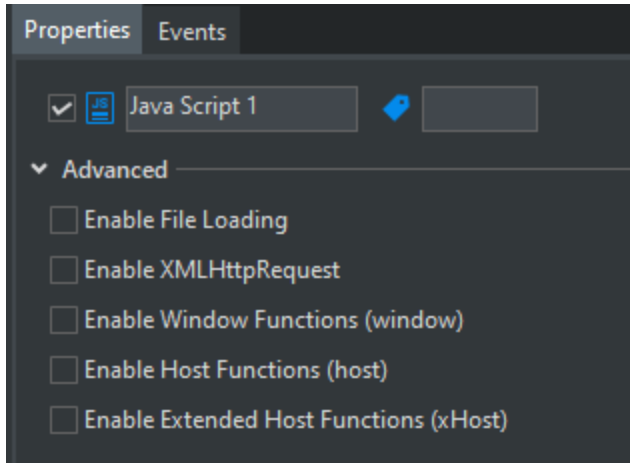


### Mode

Allows changing whether the script is run in Asynchronous mode (on a background thread), or in Synchronous mode (blocking until finished). Changing to Synchronous Mode will enable the Timeout property

### Timeout

Specifies the number of seconds that will run in Synchronous mode before timing out.



## JavaScript Advanced Options

### Enable File Loading

Allows code defined in external .js files to be imported into the script using the import keyword. The default location of module files is the Scripts folder of the Project folder belonging to the current project.

### Enable XMLHttpRequest

Allows creating a new XMLHttpRequest object that can be used for downloading data from a server.

### Enable Window Functions

Enables window functions within the script. For example alert, confirm, prompt, MsgBox and InputBox.

### Enable Host functions

Enables the host keyword within the script.  
See the HostFunctions section below for more info.

## Enable Extended Host functions

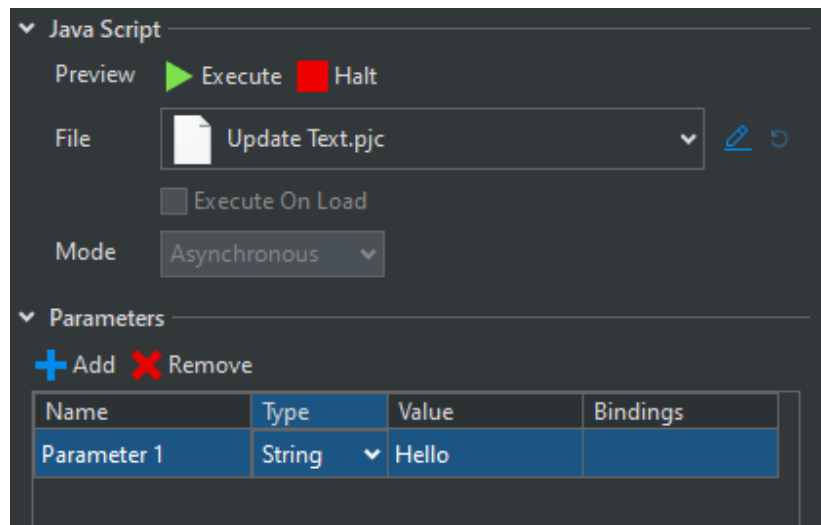
Enables the xHost keyword within the script. See the ExtendedHostFunctions section below for more info.

## Java Script Parameters

Click Add and Remove to edit the parameters of the selected JavaScript object. Any Parameters added to the JavaScript effect will be saved with the effect. All child parameter objects are natively accessible from within the script code.

### To Save a new script:



Click the disk icon and select Save or Save As.








**To import an existing JavaScript:** Navigate to script file using the File browser

▼ Java Script

Preview  Execute  Halt

File 

 Space Stats.pjc



▼  

☒ Execute On Load

Mode 

Asynchronous ▼

▼ Parameters

 Add  Remove

Name	Type	Value	Bindings
Stat Count	Integer ▼	6	
Total Height	Integer ▼	720	
Drop Note Visible	Boolean ▼	True	
Drop Note Height	Integer ▼	64	

# Keywords

The following keywords are available to use in all JavaScript objects running within the application

Keywords	Type	Description
<b>Scene</b>	Scene	Represents the scene object that the script is apart of
<b>Parent</b>	SceneObject	Represents the parent object to which the JavaScript object belongs
<b>This</b>	JavaScript	Represents the actual JavaScript object itself
<b>Project</b>	Project	Represents the project that the scene is apart of
<b>Channel</b>	Channel	Represents the channel that the scene is loaded on
<b>Playout</b>	Playout	Represents an object that can access all channels and clip players that are configured
<b>window</b>	Window	Represents an object that can be used to invoke window functions (alert, confirm, prompt, MsgBox and InputBox). Note: Enable Window Functions must be checked in the advanced section of the script
<b>host</b>	HostFunctions	Represents an object that can be used to invoke host functions within the script. Note: Enable Host Functions must be checked in the advanced section of the script
<b>xHost</b>	ExtendedHost Functions	Represents an object that can be used to invoke extended host functions within the script. Note: Enable Extended Host Functions must be checked in the advanced section of the script
<b>FindObject(string name)</b>	SceneObject	Returns the nearest object to the parent JavaScript that matches the name argument

All parameters defined in the JavaScript object are also natively available as keywords to be used within the script.

# Properties

The following types are common to object properties

## Size

Name	Type	Description
<b>Width</b>	double	Gets or sets Width value of the vector
<b>Height</b>	double	Gets or sets Height value of the vector

## Point

Name	Type	Description
<b>X</b>	double	Gets or sets X value of the point
<b>Y</b>	double	Gets or sets Y value of the point

## Vector

Name	Type	Description
<b>X</b>	double	Gets or sets X value of the vector
<b>Y</b>	double	Gets or sets Y value of the vector
<b>Z</b>	double	Gets or sets Z value of the vector
<b>Link</b>	bool	Gets or sets a value that links the axis values together proportionally. This is only used for the Scale property

## Color

Name	Type	Description
<b>R</b>	byte	Gets the red value of the color
<b>G</b>	byte	Gets the green value of the color
<b>B</b>	byte	Gets the blue value of the color
<b>A</b>	byte	Gets the alpha value of the color

Note: colors can be assigned by setting a hex code string: `text1.Color = "#FF0000"`

## Font

Name	Type	Description
<b>Name</b>	string	Gets or sets the name of the font
<b>Size</b>	double	Gets or sets size of the font
<b>Color</b>	Color	Gets or sets the color of the text object
<b>Kerning</b>	double	Gets or sets the kerning of the text object
<b>Leading</b>	double	Gets or sets the leading of the text object
<b>SpaceWidth</b>	double	Gets or sets the space width of the text object
<b>FixedPitch</b>	double	Gets or sets the fixed pitch of the text object

## Outline

Name	Type	Description
<b>Enabled</b>	bool	Gets or sets the enabled property of the outline
<b>Color</b>	Color	Gets or sets the color property of the outline
<b>Size</b>	int	Gets or sets the size of the outline

## Shadow

Name	Type	Description
<b>Enabled</b>	bool	Gets or sets the enabled property of the shadow
<b>Color</b>	Color	Gets or sets the color property of the shadow
<b>Blur</b>	int	Gets or sets the blur of the shadow
<b>Offset</b>	Vector	Gets a Vector object that represents the offset of the shadow

## TimeCode

Name	Type	Description
<b>Interlaced</b>	bool	Gets a value that represents the interlaced value of the timecode
<b>Rate</b>	double	Gets the rate of the timecode (25, 29.97, 50, 59.94, ...)
<b>TotalFrames</b>	int	Gets the total number of frames represented by the timecode
<b>TotalFields</b>	int	Gets the total number of fields represented by the timecode. This value will be the same as TotalFrames if the timecode is not interlaced
<b>TotalSeconds</b>	double	Gets the total number of seconds represented by the timecode
<b>Fields</b>	int	Gets the number of truncated fields represented by the timecode. This value will be the same as Frames if the timecode is not interlaced

<b>Frames</b>	int	Gets the number of truncated frames represented by the timecode
<b>Seconds</b>	int	Gets the number of truncated seconds represented by the timecode
<b>Minutes</b>	int	Gets the number of truncated minutes represented by the timecode
<b>Hours</b>	int	Gets the number of hours represented by the timecode
<b>Text</b>	string	Gets a string representation of the timecode in "hh:mm:ss.ff" format

## Resolution

Name	Type	Description
<b>Width</b>	double	Gets or sets Width value of the resolution
<b>Height</b>	double	Gets or sets Height value of the resolution
<b>FrameRate</b>	FrameRate	Gets an object that represents the Frame Rate of the resolution
<b>Interlaced</b>	bool	Gets a value that represents the interlaced value of the resolution

## Bindings

Name	Type	Description
<b>Count</b>	int	Gets the bindings count
<b>Text</b>	string	Gets or sets the text representation of the binding. Multiple bindings are separated by semicolons
<b>Add(string binding)</b>	void	Adds a binding to the object
<b>Remove(string binding)</b>	void	Removes a binding from the object
<b>Clear()</b>	void	Clears all bindings from the object
<b>GetTargetObject()</b>	SceneObject	Gets the first object referenced in the bindings
<b>GetTargetProperty()</b>	string	Gets the the property of the first object referenced in the bindings
<b>GetTargetValue()</b>	object	Gets the value of the property of the first object referenced in the bindings

# Objects

## SceneObject

The following properties and methods are common to all objects within the scene tree

Name	Type	Description
<b>Name</b>	string	Gets or sets the name of the scene
<b>Type</b>	string	Gets the type of the object (Text, Image, Clip,...)
<b>Tag</b>	string	Gets or sets the Tag property which can be used mark objects to be designated for custom logic
<b>Parent</b>	SceneObject	Gets the parent of the object
<b>GetValue(string property)</b>	object	Returns the value of the specified property
<b>SetValue(string property, object value)</b>	void	Sets the specified property to the value argument
<b>FindObject(string name)</b>	SceneObject	Returns the nearest object that matches the name argument

## SceneObjectList

Represents a list of child objects

Name	Type	Description
<b>Count</b>	int	Gets the count of objects in the list
<b>New(string type)</b>	SceneObject	Creates and returns a new SceneObject based on the provided type ("Text", "Image", "Clip", "Group"...). Note this object is not automatically added to the list
<b>Add(SceneObject sceneObject)</b>	void	Adds the specified object to the end of the list
<b>Insert(int index, SceneObject sceneObject)</b>	void	Inserts the specified object at the specified index location
<b>Remove(SceneObject sceneObject)</b>	bool	Removes the specified object from the list. Returns value based on success

## Scene

Scene include all fields from SceneObject and the following fields

Name	Type	Description
<b>Description</b>	string	Gets or sets the scene description
<b>Keywords</b>	string	Gets or sets the keywords for the scene

<b>Style</b>	string	Gets or sets the style of the scene
<b>Layer</b>	int	Gets or sets the layer of the scene
<b>EffectIn</b>	TriggerList	Gets the trigger list object that represents Effect In
<b>EffectOut</b>	TriggerList	Gets the trigger list object that represents Effect Out
<b>LayerIn</b>	TriggerList	Gets the trigger list object that represents Layer In
<b>LayerOut</b>	TriggerList	Gets the trigger list object that represents Layer Out
<b>PreviewIn</b>	TriggerList	Gets the trigger list object that represents Preview In
<b>Resolution</b>	Resolution	Gets the resolution of the scene
<b>FrameRate</b>	FrameRate	Gets the frame rate of the scene
<b>FilePath</b>	string	Gets the full path of the scene
<b>Directory</b>	string	Gets the directory that contains the scene file
<b>Active</b>	bool	Gets or sets value that indicates if the scene is the active scene of the channel
<b>Status</b>	string	Sets a status message that will be displayed in the status bar of the scene in the playout panel
<b>Loaded</b>	bool	Gets a value that indicates if the scene is loaded
<b>Playing</b>	bool	Gets a value that indicates if the scene is playing
<b>Updating</b>	bool	Gets a value that indicates if the scene is updated by a matching scene from preview
<b>Closed</b>	bool	Gets a value that indicates if the scene is closed
<b>Project</b>	Project	Gets the project that contains the scene
<b>Canvas</b>	Canvas	Gets the canvas object used for the display of graphics and effects.
<b>Resources</b>	SceneResources	Gets the list of resources defined for this scene such as DataObjects,
<b>Parameters</b>	ParameterList	Gets the list of parameters defined for the scene.
<b>Expressions</b>	ExpressionList	Gets the list of expressions defined for the scene.
<b>Conditions</b>	ConditionList	
<b>Replaceables</b>	ReplaceableList	
<b>Actions</b>	ActionList	Gets the list of Actions defined for this scene.
<b>AllObjects</b>	Enumerable<Scene Object>	Gets the list of all objects including graphics, controls, resources and effects that the scene contains
<b>Play()</b>	void	Plays the scene
<b>Stop()</b>	void	Stops the scene
<b>Close()</b>	void	Closes the scene
<b>Update(string name, string value)</b>	bool	Attempts to update the default value of an object matching the name argument with the value arguments. Returns true if successful, and false otherwise
<b>GetObjectValue(string name, string property)</b>	object	Gets the value of the property argument that matches an object in the scene that matches the name argument

<b>SetObjectValue(string name, string property, object value)</b>	void	Sets the value of a the property argument that matches an object in the scene that matches the name argument
<b>Execute(string command)</b>	void	Executes a command as if it was a logic statement. For instance: Scene.Execute("Image1.Opacity = 100")
<b>Evaluate(string expression)</b>	object	Evaluates an expression as if it was in a logic statement and returns the result. For instance: Scene.Evaluate("Text1.Text")
<b>QueueUpdates()</b>	QueueCommand	<p>Puts the scene into Queue Mode and returns a QueueCommand that can be used to apply all queued updates simultaneously. For example:</p> <pre>var updates = Scene.QueueUpdates();  try {     // update multiple objects in scene } finally {     updates.Execute(); }</pre>
<b>SaveMessage()</b>	string	Saves the current state of the scene to a message file in the Messages folder of the Project of the scene. The first available message number will be used as the name of the message. The path of the message file will be returned.
<b>SaveMessage(string message)</b>	string	Saves the current state of the scene to a message file in the Messages folder of the Project of the scene. The message argument can be the name of the message or the message path. The path of the message file will be returned.
<b>LoadedChanged</b>	SceneEvent	<p>Represents an event that can be connected to a function to be run every time the Loaded property changes. For example</p> <pre>Scene.LoadedChanged.connect(OnLoadedChanged);  function OnLoadedChanged(scene) {     // react to loaded changed }</pre>



<b>PlayingChanged</b>	SceneEvent	Represents an event that can be connected to a function to be run every time the Playing property changes.
<b>UpdatingChanged</b>	SceneEvent	Represents an event that can be connected to a function to be run every time the Updated property changes.
<b>StyleChanged</b>	SceneEvent	Represents an event that can be connected to a function to be run every time the Style property changes.
<b>AfterLoad</b>	SceneEvent	Represents an event that can be connected to a function to be run after the scene has finished loading.
<b>BeforePlay</b>	SceneEvent	Represents an event that can be connected to a function to be run before the scene has started playing.
<b>AfterPlay</b>	SceneEvent	Represents an event that can be connected to a function to be run after the scene has finished playing.
<b>BeforeUpdate</b>	SceneEvent	Represents an event that can be connected to a function to be run before the scene has started updating replaceable values.
<b>AfterUpdate</b>	SceneEvent	Represents an event that can be connected to a function to be run after the scene has finished updating replaceable values.
<b>BeforeStop</b>	SceneEvent	Represents an event that can be connected to a function to be run before the scene has started stopping.
<b>AfterStop</b>	SceneEvent	Represents an event that can be connected to a function to be run after the scene has finished stopping.
<b>BeforeClose</b>	SceneEvent	Represents an event that can be connected to a function to be run before the scene has started closing.

## Action

Represents a collection of animations

Name	Type	Description
<b>Name</b>	string	Gets or sets the name of the action
<b>Length</b>	TimeCode	Gets an object representing the length of the action
<b>Loop</b>	bool	Gets or sets the loop property for all of the associated animations
<b>Animations</b>	Enumerable<Animation>	Gets an object that can be used to iterate over all associated animations
<b>Play()</b>	void	Plays the animation
<b>Play(int offset)</b>	void	Plays the animation after the specified offset
<b>Stop()</b>	void	Stops the animation

## Animation

Represents a child animation of an object

Name	Type	Description
<b>Play()</b>	void	Plays the animation
<b>Play(int offset)</b>	void	Plays the animation after the specified offset
<b>Stop()</b>	void	Stops the animation
<b>Animate(SceneObject sceneObject)</b>	void	Plays the animation on the provided object
<b>Animate(SceneObject sceneObject, int offset)</b>	void	Plays the animation on the provided object after the specified offset

## AnimationList

Represents the list of child animations of an object

Name	Type	Description
<b>Default</b>	Animation	Returns the Default animation
<b>Find(string name)</b>	Animation	Returns the animation specified by the provided name if it exists, or null if not

## DynamicObject

DynamicObject includes all fields from SceneObject and the following fields

Name	Type	Description
<b>DefaultAnimation</b>	Animation	Gets or sets the Default animation
<b>Animations</b>	AnimationList	Gets an object representing the animations defined

## Graphic

Graphic includes all fields from DynamicObject and the following fields

Name	Type	Description
<b>Enabled</b>	bool	Gets or sets whether the graphic is visible or not
<b>Opacity</b>	double	Gets or sets the opacity of the graphic
<b>Position</b>	Vector	Gets a vector object that represents the Position of the graphic
<b>Scale</b>	Vector	Gets a vector object that represents the Scale of the graphic
<b>Rotation</b>	Vector	Gets a vector object that represents the Rotation of the graphic
<b>Pivot</b>	Vector	Gets a vector object that represents the Pivot of the graphic
<b>Effects</b>	SceneObject List<Effect>	Gets an object that represents the child effects of the graphic

## 2D Graphic

2D Graphic includes all fields from Graphic and the following fields

Name	Type	Description
Width	double	Gets or sets the width of the graphic
Height	double	Gets or sets the height of the graphic
Size	Size	Gets a size object that represents the size of the graphic
Origin	Point	Gets a point object that represents the origin of the graphic

## Text

Text objects include all fields from 2D Graphic and the following fields

Name	Type	Description
Text	string	Gets or sets the Text of the text object
HorizontalAlignment	HorizontalAlignment	Gets or sets the horizontal alignment of the text object: "Left", "Center", "Right", "FirstCharacter", "LastCharacter"
HorizontalAlignmentCharacter	string	
VerticalAlignment	VerticalAlignment	Gets or sets the vertical alignment of the text object: "Top", "First", "Middle", "Last", "Bottom"
WordWrap	WordWrap	Gets or sets the word wrap of the text object: "Off", "Wrap" or "UniformScale"
HorizontalScale	HorizontalScale	Gets or sets the horizontal scale of the text object: "Off", "ScaleToFit" or "KernToFit"
VerticalScale	VerticalScale	Gets or sets the vertical scale of the text object: "Off", "ScaleToFit" or "FillToFit"
AllCaps	bool	Gets or sets the all caps property of the text object
CapsRatio	double	Gets or sets the caps ratio of the text object
Numeric	bool	Gets or sets the numeric property of the text object
Number	double	Gets or sets the number property of the text object
Format	string	Gets or sets the format property of the text object when Numeric is enabled
Font	Font	Gets an object that represents the font text object containing the following properties:
• Name	string	Gets or sets the name of the font
• Size	double	Gets or sets size of the font
• Color	Color	Gets or sets the color of the text object
• Kerning	double	Gets or sets the kerning of the font
• Leading	double	Gets or sets the leading of the text object

• <b>SpaceWidth</b>	double	Gets or sets the space width of the text object
• <b>FixedPitch</b>	double	Gets or sets the fixed pitch of the text object
<b>TextMode</b>	TextMode	Gets or sets the mode of the text object (" 2d" or " 3d")
<b>_2d</b>	_2dTextProperties	Gets an object that represents the 2D text properties containing the following properties:
• <b>Resolution</b>	double	Gets or sets resolution of the 2D font
• <b>LockResolution</b>	bool	Gets or sets a value that locks the resolution to the font size
• <b>ColorMode</b>	ColorMode	Gets or sets the color mode of the 2D font ("Solid", "Linear", "Quad" or "DoubleVertical")
• <b>GradientTarget</b>	GradientTargetMode	Gets or sets the gradient target of the 2D font used for non Solid color modes ("Character", "Row" or "Object")
• <b>StartColor</b>	Color	Gets or sets the linear start color used for the Linear color mode
• <b>FinishColor</b>	Color	Gets or sets the linear finish color used for the Linear color mode
• <b>GradientAngle</b>	double	Gets or sets the angle direction of start and finish colors when ColorMode is Linear
• <b>GradientColor1</b>	Color	Gets or sets the first color used when ColorMode is set to Quad or DoubleVertical
• <b>GradientColor2</b>	Color	Gets or sets the second color used when ColorMode is set to Quad or DoubleVertical
• <b>GradientColor3</b>	Color	Gets or sets the third color used when ColorMode is set to Quad or DoubleVertical
• <b>GradientColor4</b>	Color	Gets or sets the fourth color used when ColorMode is set to Quad or DoubleVertical
• <b>Outline</b>	Outline	Gets an object that represents the Outline edge of the text object
• <b>Shadow</b>	Shadow	Gets an object that represents the Shadow edge of the text object
<b>Style</b>	string	Gets or sets a file that represents the Mode, Font, 2D and 3D properties of the text object
<b>LinkStyle</b>	bool	Gets or sets a value that represents whether the text object style is linked to the file specified in the Style property
<b>LineCount</b>	int	Gets the line count of the text object

<b>UpdateTextSize</b>	bool	Gets or sets a value that enables the TextSize property to be updated with text size as the text changes
<b>TextSize</b>	TextSize	Gets a object that represents
<b>GetTextBounds(string text)</b>	Rectangle	Returns a rectangle object that represents the bounds that the provided text would occupy
<b>TextChanged</b>	SceneObjectEvent	<p>Represents an event that can be connected to a function to be run every time the Text property changes. For example</p> <pre> var text = FindObject("Text 1");  text.TextChanged.connect(OnTextChanged);  function OnTextChanged(textObject) {     // react to text changed } </pre>

## Image

Image objects include all fields from 2D Graphic and the following fields

Name	Type	Description
<b>File</b>	string	Gets or sets the file of the image object
<b>HideOnClear</b>	bool	Gets or sets a value that causes the image to disappear when the File property set to an empty string
<b>ColorMode</b>	ColorMode	Gets or sets the color mode of the image ("Solid", "Linear", "Quad" or "DoubleVertical")
<b>Color</b>	Color	Gets or sets the color of the image object
<b>StartColor</b>	Color	Gets or sets the linear start color used for the Linear color mode
<b>FinishColor</b>	Color	Gets or sets the linear finish color used for the Linear color mode
<b>GradientAngle</b>	double	Gets or sets the angle direction of start and finish colors when ColorMode is Linear
<b>GradientColor1</b>	Color	Gets or sets the first color used when ColorMode is set to Quad or DoubleVertical
<b>GradientColor2</b>	Color	Gets or sets the second color used when ColorMode is set to Quad or DoubleVertical
<b>GradientColor3</b>	Color	Gets or sets the third color used when ColorMode is set to Quad or DoubleVertical
<b>GradientColor4</b>	Color	Gets or sets the fourth color used when ColorMode is set to Quad or DoubleVertical
<b>Shadow</b>	Shadow	Gets an object that represents the Shadow edge of the imageobject
<b>FileChanged</b>	SceneObjectEvent	Represents an event that can connected to a function to be run every time the File property changes

## Clip

Clip objects include all fields from 2D Graphic and the following fields

Name	Type	Description
<b>File</b>	string	Gets or sets the file of the clip
<b>Color</b>	Color	Gets or sets the color of the clip
<b>Loop</b>	bool	Gets or sets the loop value of the clip
<b>LoopCount</b>	int	Gets or sets the count for looping
<b>Speed</b>	double	Gets or sets the speed of the clip
<b>Volume</b>	double	Gets or sets the volume of the clip
<b>Muted</b>	bool	Gets or sets whether the clip is muted
<b>HoldLastFrame</b>	bool	Gets or sets a value that controls the behavior that occurs after the clip is finished: the last frame is held if true, or clip goes if false
<b>Length</b>	TimeCode	Gets an object representing the length of the clip
<b>CurrentFrame</b>	TimeCode	Gets an object representing the current frame of the clip
<b>Cue()</b>	void	Sets the clip to the first frame
<b>Play()</b>	void	Plays the clip from the beginning
<b>Stop()</b>	void	Stops and hides the clip
<b>Pause()</b>	void	Pauses the clip at the current frame
<b>Resume()</b>	void	Resumes the clip from the current frame
<b>End()</b>	void	Sets the clip to the last frame
<b>Jog(int frames)</b>	void	Offsets the clip by the specified number of frames
<b>Mute()</b>	void	Mutes the clip
<b>Unmute()</b>	void	Unmutes the clip
<b>FileChanged</b>	SceneObjectEvent	Represents an event that can connected to a function to be run every time the File property changes



## Group

Group objects include all fields from Graphic and the following fields

Name	Type	Description
<b>Graphics</b>	SceneObjectList<Graphic>	Gets an object representing the list of child graphics
<b>ObjectsRecursive</b>	Enumerable<SceneObject>	<p>Gets an object that can be used to iterate over all children objects recursively. This will include both graphics and effects. For example:</p> <pre>var sceneGroup = Scene.SceneGroup;  for (var object in sceneGroup.ObjectsRecursive) {     // operate on object }</pre>

## Control

Controls include all fields from Scene Object and the following fields

Name	Type	Description
<b>Enabled</b>	bool	Gets or sets the enabled property of the control
<b>Visible</b>	bool	Gets or sets the visible property of the control
<b>X</b>	int	Gets or sets the X position of the control
<b>Y</b>	int	Gets or sets the Y position of the control
<b>Width</b>	int	Gets or sets the width of the control
<b>Height</b>	int	Gets or sets the height of the control
<b>Location</b>	Point	Gets the location of the control
<b>Size</b>	Size	Gets the size of the control
<b>BackColor</b>	Color	Gets the back color of the control
<b>ForeColor</b>	Color	
<b>Text</b>	string	Gets the text of the control
<b>Bindings</b>	Bindings	Gets the bindings of the control
<b>GetTargetObject()</b>	SceneObject	Gets the first object referenced in the bindings
<b>GetTargetProperty()</b>	string	Gets the the property of the first object referenced in the bindings
<b>GetTargetValue()</b>	object	Gets the value of the property of the first object referenced in the bindings
<b>TextChanged</b>	SceneObjectEvent	Represents an event that can be connected to a function to be run every time the Text property changes

## Control Panel

Control Panels include all fields from Control and the following fields

Name	Type	Description
<b>ActiveControl</b>	Control	Gets or sets the active control of the Control Panel
<b>Controls</b>	SceneObjectList<Controls>	Gets an object representing the list of child controls

## Button

Buttons include all fields from Control and the following fields

Name	Type	Description
<b>Click</b>	SceneObjectEvent	<p>Represents an event that can be connected to a function to be run every time the button is clicked. For example</p> <pre>var button = FindObject("Button 1");  button.Click.connect(OnButtonClicked);  function OnButtonClicked(buttonObject) {     // react to button clicked }</pre>

## Check Box

Check boxes include all fields from Control and the following fields

Name	Type	Description
<b>Checked</b>	bool	Gets or sets the Checked property of the text box
<b>CheckedChanged</b>	SceneObjectEvent	Represents an event that can be connected to a function to be run every time the Checked property changes

## Text Box

Text boxes include all fields from Control and the following fields

Name	Type	Description
<b>Multiline</b>	bool	Gets or sets the Multiline property of the text box
<b>SelectedText</b>	string	Gets or sets the selected text of the text box
<b>SelectionStart</b>	int	Gets or sets the beginning index of the text box selection
<b>SelectionLength</b>	int	Gets or sets the length of the text box selection

## Combo Box

Combo boxes include all fields from Control and the following fields

Name	Type	Description
<b>Items</b>	ComboBoxItems	Gets an object representing combo box items

## Parameter

Includes all fields from SceneObject plus the following fields

Name	Type	Description
<b>Enabled</b>	bool	Gets or sets a value representing the enabled property of the parameter
<b>Value</b>	object	Gets or sets the value of the parameter
<b>PreviousValue</b>	object	Gets the previous value of the parameter
<b>Bindings</b>	Bindings	Gets the bindings for the parameter
<b>ValueChanged</b>	SceneObjectEvent	<p>Represents an event that can be connected to a function to be run every time the Value property changes. For example</p> <pre>var text = FindObject("Text 1");  Parameter1.ValueChanged.connect(OnValueChanged);  function OnValueChanged(parameter) {     // react to parameter value changed }</pre>

## JavaScript

Includes all fields from SceneObject plus the following fields

Name	Type	Description
<b>Sync</b>	bool	Gets or sets the Sync property of the script that determines whether the script is run synchronously or asynchronously. For example, to enable:  This.Sync = true;
<b>Timeout</b>	double	Gets or sets the timeout in seconds that is used to wait for synchronous scripts to finish. The default value is 1 second
<b>Parameters</b>	ParametersList	Gets an object representing the list of child parameters

## Project

Includes all fields from Scene plus the following fields

Name	Type	Description
<b>Path</b>	string	Gets the directory path of the Project

## Channel

Represents an object that can be used to affect the scenes loaded and played in a channel

Name	Type	Description
<b>Index</b>	int	Gets the 0 based index of the channel as defined by the order in the playout configuration
<b>Number</b>	int	Gets the 1 based index of the channel as defined by the order in the playout configuration
<b>Name</b>	string	Gets the name of the channel as defined by the order in the playout configuration
<b>Resolution</b>	Size	Gets the resolution of the channel as defined by the playout configuration
<b>ActivePreviewScene</b>	Scene	Gets the active scene in preview
<b>ActiveProgramScene</b>	Scene	Gets the active scene in program
<b>OpenScenes</b>	Enumerable<Scene>	Gets an object that can be used to iterate over all of the open scenes in the channel

<b>GetScene(string name)</b>	Scene	Returns the scene in the channel that matches the name argument if it exists, or null if not
<b>LoadScene(string name)</b>	bool	Loads the specified scene name into the channel. Returns the success of finding the scene. Note: the name can be a file name within the current project, or a full file path
<b>PlayScene(string name)</b>	bool	Plays the specified scene name on the channel. Returns the success of finding the scene
<b>StopScene(string name)</b>	bool	Stops the specified scene name on the channel. Returns the success of finding the scene
<b>CloseScene(string name)</b>	bool	Closes the specified scene name on the channel. Returns the success of finding the scene
<b>RenderImage(string path = null)</b>	string	Renders the playing scene(s) on a Render channel to an image file. An optional path parameter can be passed as an argument. Returns the location of the image file

## ClipPlayer

Represents an object that can be used to affect the clips cued and played in a clip player

Name	Type	Description
<b>Index</b>	int	Gets the 0 based index of the clip player as defined by the order in the playout configuration
<b>Number</b>	int	Gets the 1 based index of the clip player as defined by the order in the playout configuration
<b>Name</b>	string	Gets the name of the clip player as defined by the order in the playout configuration
<b>CuedClip</b>	Scene	Gets the cued clip scene
<b>PlayingClip</b>	Scene	Gets the playing clip scene
<b>ClipExists(string name)</b>	bool	Returns true if the specified clip name exists within the project, false if not
<b>CueClip(string name)</b>	bool	Loads the specified scene file into the channel. Returns the success of finding the scene. Note: the file can be a file name within the current project, or a full file path
<b>PlayCued()</b>	void	Plays the current cued clip
<b>ClearCued()</b>	void	Clears the current cued clip

<b>PausePlaying()</b>	void	Pauses the playing clip
<b>ResumePlaying()</b>	void	Resumes the playing clip
<b>StopPlaying()</b>	void	Stops the playing clip
<b>ClearPlaying()</b>	void	Clears the playing clip
<b>JogPlaying(int frames)</b>	void	Jogs the playing clip by the specified number of frames

## Playout

Represents an object that can access all channels and clip players that are configured

Name	Type	Description
<b>Channels</b>	ReadOnlyList<Channel>	Gets a read only list that represents the channels defined in the current playout configuration
<b>ClipPlayers</b>	ReadOnlyList<ClipPlayer>	Gets a read only list that represents the clip players defined in the current playout configuration
<b>ChannelExists(int number)</b>	bool	Returns true if a channel exists at this number
<b>ChannelExists(string name)</b>	bool	Returns true if a channel exists at this name
<b>GetChannel(int number)</b>	Channel	Returns the channel object at the specified number
<b>GetChannel(string name)</b>	Channel	Returns the channel object at the specified name
<b>ClipPlayerExists(int number)</b>	bool	Returns true if a clip player exists at this number
<b>ClipPlayerExists(string name)</b>	bool	Returns true if a clip player exists at this name
<b>GetClipPlayer(int number)</b>	ClipPlayer	Returns the clip player object at the specified number
<b>GetClipPlayer(string name)</b>	ClipPlayer	Returns the clip player object at the specified name

## Window

Represents an object that can be used to display popup message boxes

Name	Type	Description
<b>alert(string message)</b>	void	Displays an OK message box with the provided message text  <a href="https://developer.mozilla.org/en-US/docs/Web/API/Window/alert">https://developer.mozilla.org/en-US/docs/Web/API/Window/alert</a>
<b>confirm(string message)</b>	bool	Displays an OK/Cancel message box with the provided message text. Returns true if OK is clicked and false if Cancel is clicked  <a href="https://developer.mozilla.org/en-US/docs/Web/API/Window/confirm">https://developer.mozilla.org/en-US/docs/Web/API/Window/confirm</a>
<b>prompt(string message, string defaultValue = "")</b>	string	Displays an OK/Cancel input box with the provided message text and optional default input value. Returns the input text if OK is clicked, or blank if Cancel is clicked  <a href="https://developer.mozilla.org/en-US/docs/Web/API/Window/prompt">https://developer.mozilla.org/en-US/docs/Web/API/Window/prompt</a>
<b>MsgBox(string message)</b>	void	Displays an OK message box with the provided message text
<b>InputDialog(string message, string title = "", string defaultValue = "", int x = -1, int y = -1)</b>	string	Displays an OK/Cancel input box with the provided message text, optional title and optional default input value. Returns the input text if OK is clicked, or blank if Cancel is clicked

# HostFunctions

The HostFunctions object provides access to basic .NET functionality as defined here:

[https://microsoft.github.io/ClearScript/Reference/html/T\\_Microsoft\\_ClearScript\\_HostFunctions.htm](https://microsoft.github.io/ClearScript/Reference/html/T_Microsoft_ClearScript_HostFunctions.htm)

For example, a .NET array can be created if necessary:

```
var array = host.newArr(10);
```

## ExtendedHostFunctions

The ExtendedHostFunctions object provides access to extended .NET functionality as defined here:

[https://microsoft.github.io/ClearScript/Reference/html/T\\_Microsoft\\_ClearScript\\_HostFunctions.htm](https://microsoft.github.io/ClearScript/Reference/html/T_Microsoft_ClearScript_HostFunctions.htm)

For example, the .NET File class can be referenced to access file information on disk:

```
var File = xHost.type("System.IO.File");  
  
var text = File.ReadAllText(filePath);
```

Other libraries can be accessed for specific classes:

```
var DrawingLib = xHost.lib("System.Drawing");  
  
var Color = DrawingLib.System.Drawing.Color;
```



## **ABOUT US**

Chyron is ushering in the next generation of storytelling in the digital age. Founded in 1966, the company pioneered broadcast titling and graphics systems. With a strong foundation built on over 50 years of innovation and efficiency, the name Chyron is synonymous with broadcast graphics. Chyron continues that legacy as a global leader focused on customer-centric broadcast solutions. Today, the company offers production professionals the industry's most comprehensive software portfolio for designing, sharing, and playing live graphics to air with ease. Chyron products are increasingly deployed to empower OTA & OTT workflows and deliver richer, more immersive experiences for audiences and sports fans in the arena, at home, or on the go.

## **CONTACT SALES**

EMEA • North America • Latin America • Asia/Pacific  
+1.631.845.2000 • [sales@chyron.com](mailto:sales@chyron.com)

