JSFile

Method Summary

Boolean	canRead() Returns true if the file exists and is readable (has access to it) - works on remote files too.
Boolean	canWrite() Returns true if the file exists and can be modified - works on remote files too.
Boolean	createNewFile() Creates the file on disk if needed.
Boolean	deleteFile() Deletes the file from the disk if possible.
Boolean	exists() Returns true if the file/directory exists on the filesystem - works on remote files too.
JSFile	getAbsoluteFile() Returns a JSFile instance that corresponds to the absolute form of this pathname - works on remote files too.
String	getAbsolutePath() Returns a String representation of the absolute form of this pathname - works on remote files too.
byte[]	getBytes() Gets the contents (bytes) for the file data.
String	getContentType() Returns the contenttype of this file, like for example 'application/pdf' - works on remote files too.
String	getName() Returns the name of the file.
String	getParent() Returns the String representation of the path of the parent of this file - works on remote files too.
JSFile	getParentFile() Returns a JSFile instance that corresponds to the parent of this file - works on remote files too.
String	getPath() Returns a String holding the path to the file - works on remote files too.
Boolean	isAbsolute() Returns true if the path is absolute.
Boolean	isDirectory() Returns true if the file is a directory - works on remote files too.
Boolean	isFile() Returns true if the file is a file and not a regular file - works on remote files too.
Boolean	isHidden() Returns true if the file is hidden (a file system attribute) - works on remote files too.
Date	lastModified() Returns the time/date of the last modification on the file - works on remote files too.
String[]	list() Returns an array of strings naming the files and directories located inside the file, if the file is a directory - works on remote files too.
JSFile[]	listFiles() Returns an array of JSFiles naming the files and directories located inside the file, if the file is a directory - works on remote files too.
Boolean	mkdir() Creates a directory on disk if possible.
Boolean	mkdirs() Creates a directory on disk, together with all its parent directories, if possible.
Boolean	renameTo(destination) Renames the file to a different name.
Boolean	setBytes(bytes) Set the content of the file (local or remote) to the bytes provided Will not create a new file if one doesn't exist
Boolean	setBytes(bytes, createFile) Set the content of the file (local or remote) to the bytes provided
Boolean	setLastModified(date) Sets the date/time of the last modification on the file.
Boolean	setReadOnly() Sets the readonly attribute of the file/directory.
Number	size() Returns the size in bytes of the file.

Method Details

canRead

Boolean **canRead** ()

Returns true if the file exists and is readable (has access to it) - works on remote files too.

Returns

Boolean

Sample

```
var f = plugins.file.convertToJSFile('./big.jpg');
// or for a remote file:
// var f = plugins.convertToRemoteJSFile('/images/big.jpg');
if (f && f.exists()) {
        application.output('is absolute: ' + f.isAbsolute());
        application.output('is dir: ' + f.isDirectory());
        application.output('is file: ' + f.isFile());
        application.output('is hidden: ' + f.isHidden());
        application.output('can read: ' + f.canRead());
        application.output('can write: ' + f.canWrite());
        application.output('last modified: ' + f.lastModified());
        application.output('name: ' + f.getName());
        application.output('path: ' + f.getPath());
        application.output('absolute path: ' + f.getAbsolutePath());
        application.output('content type: ' + f.getContentType());
        application.output('size: ' + f.size());
}
else {
        application.output('File/folder not found.');
}
```

canWrite

Boolean **canWrite** ()

Returns true if the file exists and can be modified - works on remote files too.

Returns

Boolean

Sample

```
var f = plugins.file.convertToJSFile('./big.jpg');
// or for a remote file:
// var f = plugins.convertToRemoteJSFile('/images/big.jpg');
if (f && f.exists()) {
        application.output('is absolute: ' + f.isAbsolute());
        application.output('is dir: ' + f.isDirectory());
        application.output('is file: ' + f.isFile());
        application.output('is hidden: ' + f.isHidden());
        application.output('can read: ' + f.canRead());
        application.output('can write: ' + f.canWrite());
        application.output('last modified: ' + f.lastModified());
        application.output('name: ' + f.getName());
        application.output('path: ' + f.getPath());
        application.output('absolute path: ' + f.getAbsolutePath());
        application.output('content type: ' + f.getContentType());
        application.output('size: ' + f.size());
}
else {
        application.output('File/folder not found.');
}
```

createNewFile

Boolean **createNewFile** ()

Creates the file on disk if needed. Returns true if the file (name) did not already exists and had to be created - for remote, use the streamFilesToServer to stream a file.

Returns

Boolean

Sample

```
var f = plugins.file.convertToJSFile('story.txt');
if (!f.exists())
        f.createNewFile();
```

deleteFile

Boolean **deleteFile** ()

Deletes the file from the disk if possible. Returns true if the file could be deleted. If the file is a directory, then it must be empty in order to be deleted - works on remote files too.

Returns

Boolean

Sample

```
var f = plugins.file.convertToJSFile('story.txt');
// or for a remote file:
// var f = plugins.convertToRemoteJSFile('/story.txt');
if (f && f.exists())
        f.deleteFile();
```

exists

Boolean **exists** ()

Returns true if the file/directory exists on the filesystem - works on remote files too.

Returns

Boolean

Sample

```
var f = plugins.file.convertToJSFile('./big.jpg');
// or for a remote file:
// var f = plugins.convertToRemoteJSFile('/images/big.jpg');
if (f && f.exists()) {
        application.output('is absolute: ' + f.isAbsolute());
        application.output('is dir: ' + f.isDirectory());
        application.output('is file: ' + f.isFile());
        application.output('is hidden: ' + f.isHidden());
        application.output('can read: ' + f.canRead());
        application.output('can write: ' + f.canWrite());
        application.output('last modified: ' + f.lastModified());
        application.output('name: ' + f.getName());
        application.output('path: ' + f.getPath());
        application.output('absolute path: ' + f.getAbsolutePath());
        application.output('content type: ' + f.getContentType());
        application.output('size: ' + f.size());
}
else {
        application.output('File/folder not found.');
}
```

getAbsoluteFile

JSFile **getAbsoluteFile** ()

Returns a JSFile instance that corresponds to the absolute form of this pathname - works on remote files too.

Returns

JSFile

Sample

```
var f = plugins.file.convertToJSFile('story.txt');
// or for a remote file:
// var f = plugins.file.convertToRemoteJSFile('/story.txt');
application.output('parent folder: ' + f.getAbsoluteFile().getParent());
application.output('parent folder has ' + f.getAbsoluteFile().getParentFile().listFiles().length + '
entries');
```

getAbsolutePath

String **getAbsolutePath** ()

Returns a String representation of the absolute form of this pathname - works on remote files too.

Returns

String

Sample

```
var f = plugins.file.convertToJSFile('./big.jpg');
// or for a remote file:
// var f = plugins.convertToRemoteJSFile('/images/big.jpg');
if (f && f.exists()) {
        application.output('is absolute: ' + f.isAbsolute());
        application.output('is dir: ' + f.isDirectory());
        application.output('is file: ' + f.isFile());
        application.output('is hidden: ' + f.isHidden());
        application.output('can read: ' + f.canRead());
        application.output('can write: ' + f.canWrite());
        application.output('last modified: ' + f.lastModified());
        application.output('name: ' + f.getName());
        application.output('path: ' + f.getPath());
        application.output('absolute path: ' + f.getAbsolutePath());
        application.output('content type: ' + f.getContentType());
        application.output('size: ' + f.size());
}
else {
        application.output('File/folder not found.');
}
```

getBytes

byte[] **getBytes** ()

Gets the contents (bytes) for the file data.

Returns

byte[]

Sample

```
var theFile = plugins.file.showFileOpenDialog();
application.output('The file size in bytes: ' + theFile.getBytes());
```

getContentType

String **getContentType** ()

Returns the contenttype of this file, like for example 'application/pdf' - works on remote files too.

Returns

String

Sample

```
var f = plugins.file.convertToJSFile('./big.jpg');
// or for a remote file:
// var f = plugins.convertToRemoteJSFile('/images/big.jpg');
if (f && f.exists()) {
        application.output('is absolute: ' + f.isAbsolute());
        application.output('is dir: ' + f.isDirectory());
        application.output('is file: ' + f.isFile());
        application.output('is hidden: ' + f.isHidden());
        application.output('can read: ' + f.canRead());
        application.output('can write: ' + f.canWrite());
        application.output('last modified: ' + f.lastModified());
        application.output('name: ' + f.getName());
        application.output('path: ' + f.getPath());
        application.output('absolute path: ' + f.getAbsolutePath());
        application.output('content type: ' + f.getContentType());
        application.output('size: ' + f.size());
}
else {
        application.output('File/folder not found.');
}
```

getName

String **getName** ()

Returns the name of the file. The name consists in the last part of the file path - works on remote files too.

Returns

String

Sample

```
var f = plugins.file.convertToJSFile('./big.jpg');
// or for a remote file:
// var f = plugins.convertToRemoteJSFile('/images/big.jpg');
if (f && f.exists()) {
        application.output('is absolute: ' + f.isAbsolute());
        application.output('is dir: ' + f.isDirectory());
        application.output('is file: ' + f.isFile());
        application.output('is hidden: ' + f.isHidden());
        application.output('can read: ' + f.canRead());
        application.output('can write: ' + f.canWrite());
        application.output('last modified: ' + f.lastModified());
        application.output('name: ' + f.getName());
        application.output('path: ' + f.getPath());
        application.output('absolute path: ' + f.getAbsolutePath());
        application.output('content type: ' + f.getContentType());
        application.output('size: ' + f.size());
}
else {
        application.output('File/folder not found.');
}
```

getParent

String **getParent** ()

Returns the String representation of the path of the parent of this file - works on remote files too.

Returns

String

Sample

```
var f = plugins.file.convertToJSFile('story.txt');
// or for a remote file:
// var f = plugins.file.convertToRemoteJSFile('/story.txt');
application.output('parent folder: ' + f.getAbsoluteFile().getParent());
application.output('parent folder has ' + f.getAbsoluteFile().getParentFile().listFiles().length + '
entries');
```

getParentFile

JSFile **getParentFile** ()

Returns a JSFile instance that corresponds to the parent of this file - works on remote files too.

Returns

JSFile

Sample

```
var f = plugins.file.convertToJSFile('story.txt');
// or for a remote file:
// var f = plugins.file.convertToRemoteJSFile('/story.txt');
application.output('parent folder: ' + f.getAbsoluteFile().getParent());
application.output('parent folder has ' + f.getAbsoluteFile().getParentFile().listFiles().length + '
entries');
```

getPath

String **getPath** ()

Returns a String holding the path to the file - works on remote files too.

Returns

String

Sample

```
var f = plugins.file.convertToJSFile('./big.jpg');
// or for a remote file:
// var f = plugins.convertToRemoteJSFile('/images/big.jpg');
if (f && f.exists()) {
        application.output('is absolute: ' + f.isAbsolute());
        application.output('is dir: ' + f.isDirectory());
        application.output('is file: ' + f.isFile());
        application.output('is hidden: ' + f.isHidden());
        application.output('can read: ' + f.canRead());
        application.output('can write: ' + f.canWrite());
        application.output('last modified: ' + f.lastModified());
        application.output('name: ' + f.getName());
        application.output('path: ' + f.getPath());
        application.output('absolute path: ' + f.getAbsolutePath());
        application.output('content type: ' + f.getContentType());
        application.output('size: ' + f.size());
}
else {
        application.output('File/folder not found.');
}
```

isAbsolute

Boolean **isAbsolute** ()

Returns true if the path is absolute. The path is absolute if it starts with '/' on Unix/Linux/MacOS or has a driver letter on Windows - works on remote files too.

Returns

Boolean

Sample

```
var f = plugins.file.convertToJSFile('./big.jpg');
// or for a remote file:
// var f = plugins.convertToRemoteJSFile('/images/big.jpg');
if (f && f.exists()) {
        application.output('is absolute: ' + f.isAbsolute());
        application.output('is dir: ' + f.isDirectory());
        application.output('is file: ' + f.isFile());
        application.output('is hidden: ' + f.isHidden());
        application.output('can read: ' + f.canRead());
        application.output('can write: ' + f.canWrite());
        application.output('last modified: ' + f.lastModified());
        application.output('name: ' + f.getName());
        application.output('path: ' + f.getPath());
        application.output('absolute path: ' + f.getAbsolutePath());
        application.output('content type: ' + f.getContentType());
        application.output('size: ' + f.size());
}
else {
        application.output('File/folder not found.');
}
```

isDirectory

Boolean **isDirectory** ()

Returns true if the file is a directory - works on remote files too.

Returns

Boolean

Sample

```
var f = plugins.file.convertToJSFile('./big.jpg');
// or for a remote file:
// var f = plugins.convertToRemoteJSFile('/images/big.jpg');
if (f && f.exists()) {
        application.output('is absolute: ' + f.isAbsolute());
        application.output('is dir: ' + f.isDirectory());
        application.output('is file: ' + f.isFile());
        application.output('is hidden: ' + f.isHidden());
        application.output('can read: ' + f.canRead());
        application.output('can write: ' + f.canWrite());
        application.output('last modified: ' + f.lastModified());
        application.output('name: ' + f.getName());
        application.output('path: ' + f.getPath());
        application.output('absolute path: ' + f.getAbsolutePath());
        application.output('content type: ' + f.getContentType());
        application.output('size: ' + f.size());
}
else {
        application.output('File/folder not found.');
}
```

isFile

Boolean **isFile** ()

Returns true if the file is a file and not a regular file - works on remote files too.

Returns

Boolean

Sample

```
var f = plugins.file.convertToJSFile('./big.jpg');
// or for a remote file:
// var f = plugins.convertToRemoteJSFile('/images/big.jpg');
if (f && f.exists()) {
        application.output('is absolute: ' + f.isAbsolute());
        application.output('is dir: ' + f.isDirectory());
        application.output('is file: ' + f.isFile());
        application.output('is hidden: ' + f.isHidden());
        application.output('can read: ' + f.canRead());
        application.output('can write: ' + f.canWrite());
        application.output('last modified: ' + f.lastModified());
        application.output('name: ' + f.getName());
        application.output('path: ' + f.getPath());
        application.output('absolute path: ' + f.getAbsolutePath());
        application.output('content type: ' + f.getContentType());
        application.output('size: ' + f.size());
}
else {
        application.output('File/folder not found.');
}
```

isHidden

Boolean **isHidden** ()

Returns true if the file is hidden (a file system attribute) - works on remote files too.

Returns

Boolean

Sample

```
var f = plugins.file.convertToJSFile('./big.jpg');
// or for a remote file:
// var f = plugins.convertToRemoteJSFile('/images/big.jpg');
if (f && f.exists()) {
        application.output('is absolute: ' + f.isAbsolute());
        application.output('is dir: ' + f.isDirectory());
        application.output('is file: ' + f.isFile());
        application.output('is hidden: ' + f.isHidden());
        application.output('can read: ' + f.canRead());
        application.output('can write: ' + f.canWrite());
        application.output('last modified: ' + f.lastModified());
        application.output('name: ' + f.getName());
        application.output('path: ' + f.getPath());
        application.output('absolute path: ' + f.getAbsolutePath());
        application.output('content type: ' + f.getContentType());
        application.output('size: ' + f.size());
}
else {
        application.output('File/folder not found.');
}
```

lastModified

Date **lastModified** ()

Returns the time/date of the last modification on the file - works on remote files too.

Returns

Date

Sample

```
var f = plugins.file.convertToJSFile('./big.jpg');
// or for a remote file:
// var f = plugins.convertToRemoteJSFile('/images/big.jpg');
if (f && f.exists()) {
        application.output('is absolute: ' + f.isAbsolute());
        application.output('is dir: ' + f.isDirectory());
        application.output('is file: ' + f.isFile());
        application.output('is hidden: ' + f.isHidden());
        application.output('can read: ' + f.canRead());
        application.output('can write: ' + f.canWrite());
        application.output('last modified: ' + f.lastModified());
        application.output('name: ' + f.getName());
        application.output('path: ' + f.getPath());
        application.output('absolute path: ' + f.getAbsolutePath());
        application.output('content type: ' + f.getContentType());
        application.output('size: ' + f.size());
}
else {
        application.output('File/folder not found.');
}
```

list

String[] **list** ()

Returns an array of strings naming the files and directories located inside the file, if the file is a directory - works on remote files too.

Returns

String[]

Sample

```
var d = plugins.file.convertToJSFile('plugins');
// or for a remote file:
// var d = plugins.convertToRemoteJSFile('/plugins');
var names = d.list();
application.output('Names:');
for (var i=0; i<names.length; i++)
        application.output(names[i]);
var files = d.listFiles();
application.output('Absolute paths:');
for (var i=0; i<files.length; i++)
        application.output(files[i].getAbsolutePath());
```

listFiles

JSFile[] **listFiles** ()

Returns an array of JSFiles naming the files and directories located inside the file, if the file is a directory - works on remote files too.

Returns

JSFile[]

Sample

```
var d = plugins.file.convertToJSFile('plugins');
// or for a remote file:
// var d = plugins.convertToRemoteJSFile('/plugins');
var names = d.list();
application.output('Names:');
for (var i=0; i<names.length; i++)
        application.output(names[i]);
var files = d.listFiles();
application.output('Absolute paths:');
for (var i=0; i<files.length; i++)
        application.output(files[i].getAbsolutePath());
```

mkdir

Boolean **mkdir** ()

Creates a directory on disk if possible. Returns true if a new directory was created - for remote, use the streamFilesToServer to create the directory instead.

Returns

Boolean

Sample

```
var f = plugins.file.convertToJSFile('one/two/three/four');
f.mkdirs(); // Create all four levels of folders in one step.
var g = plugins.file.convertToJSFile('one/two/three/four/five');
g.mkdir(); // This will work because all parent folders are already created.
```

mkdirs

Boolean **mkdirs** ()

Creates a directory on disk, together with all its parent directories, if possible. Returns true if the hierarchy of directories is created - for remote, use the streamFilesToServer to create the directories instead.

Returns

Boolean

Sample

```
var f = plugins.file.convertToJSFile('one/two/three/four');
f.mkdirs(); // Create all four levels of folders in one step.
var g = plugins.file.convertToJSFile('one/two/three/four/five');
g.mkdir(); // This will work because all parent folders are already created.
```

renameTo

Boolean **renameTo** (destination)

Renames the file to a different name. Returns true if the file could be renamed - works on remote files too.

Parameters

{Object} destination

Returns

Boolean

Sample

```
var f = plugins.file.convertToJSFile('story.txt');
f.renameTo('otherstory.txt');
// or for a remote file:
// var f = plugins.convertToRemoteJSFile('/story.txt');
// f.renameTo('/otherstory.txt');
```

setBytes

Boolean **setBytes** (bytes)

Set the content of the file (local or remote) to the bytes provided
Will not create a new file if one doesn't exist

Since

5.2.5

Parameters

{byte[]} bytes - the data

Returns

Boolean - true if the operation worked

Sample

```
var file = plugins.file.convertToJSFile('/pathTo/file.jpg');
// or for a remote file:
// var file = plugins.file.convertToRemoteJSFile('/remotePathTo/file.jpg');
var success = file.setBytes(blobDataProvider, true);
```

setBytes

Boolean **setBytes** (bytes, createFile)

Set the content of the file (local or remote) to the bytes provided

Since

5.2.5

Parameters

{byte[]} bytes - the data
{Boolean} createFile - true to create a file if it doesn't exist

Returns

Boolean - true if the operation worked

Sample

```
var file = plugins.file.convertToJSFile('/pathTo/file.jpg');
// or for a remote file:
// var file = plugins.file.convertToRemoteJSFile('/remotePathTo/file.jpg');
var success = file.setBytes(blobDataProvider, true);
```

setLastModified

Boolean **setLastModified** (date)

Sets the date/time of the last modification on the file.

Parameters

{Object} date

Returns

Boolean

Sample

```
var f = plugins.file.convertToJSFile('story.txt');
f.createNewFile();
// Make the file look old.
f.setLastModified(new Date(1999, 5, 21));
```

setReadOnly

Boolean **setReadOnly** ()

Sets the readonly attribute of the file/directory. Returns true on success.

Returns

Boolean

Sample

```
var f = plugins.file.convertToJSFile('invoice.txt');
plugins.file.writeTXTFile(f, 'important data that should not be changed');
f.setReadOnly();
```

size

Number **size** ()

Returns the size in bytes of the file. Returns 0 if the file does not exist on disk - works on remote files too.

Returns

Number

Sample

```
var f = plugins.file.convertToJSFile('./big.jpg');
// or for a remote file:
// var f = plugins.convertToRemoteJSFile('/images/big.jpg');
if (f && f.exists()) {
        application.output('is absolute: ' + f.isAbsolute());
        application.output('is dir: ' + f.isDirectory());
        application.output('is file: ' + f.isFile());
        application.output('is hidden: ' + f.isHidden());
        application.output('can read: ' + f.canRead());
        application.output('can write: ' + f.canWrite());
        application.output('last modified: ' + f.lastModified());
        application.output('name: ' + f.getName());
        application.output('path: ' + f.getPath());
        application.output('absolute path: ' + f.getAbsolutePath());
        application.output('content type: ' + f.getContentType());
        application.output('size: ' + f.size());
}
else {
        application.output('File/folder not found.');
}
```