

JSProgressMonitor

Method Summary

Boolean	#cancel() Cancels the transfer process.
Number	#getCurrentBytesToTransfer() Returns the number of bytes to transfer for the current file.
Number	#getCurrentFileIndex() Returns the index of the current file being transferred.
Number	#getCurrentTransferredBytes() Returns the number of bytes already transferred for the current file.
String	#getCurrentTransferredFileName() Returns the name of the current file being transferred.
Number	#getTotalBytesToTransfer() Returns the total bytes to transfer to or from the server (sum of all the files size)
Number	#getTotalFilesToTransfer() Returns the total number of files to transfer.
Number	#getTotalTransferredBytes() Returns the total bytes already transferred (for all files)
Boolean	#isCanceled() Returns true if the process was canceled.
Boolean	#isFinished() Returns true if the process is finished.
JSProgressMonitor	#setProgressCallBack(progressCallbackFunction, interval, [testDelay]) Sets a method to be called repeatedly at the given interval (in seconds), the method will receive an instance of this JSProgressMonitor updated with the latest values.

Method Details

cancel

Boolean **cancel()**

Cancels the transfer process.

Returns

Boolean

Sample

```
monitor.cancel();
```

getCurrentBytesToTransfer

Number **getCurrentBytesToTransfer()**

Returns the number of bytes to transfer for the current file.

Returns

Number

Sample

```
application.output('total transferred: ' + monitor.getTotalTransferredBytes() + ' / ' + monitor.
getTotalBytesToTransfer());
    application.output('current file: ' + monitor.getCurrentTransferredFileName() + ' ( ' + monitor.
getCurrentFileIndex() + ' / ' + monitor.getTotalFilesToTransfer() + ' )');
    application.output('current bytes transferred: '+monitor.getCurrentTransferredBytes() + ' / ' + monitor.
getCurrentBytesToTransfer());
    if (monitor.isCanceled()) {
        application.output('canceled!')
    }
    if (monitor.isFinished()) {
        application.output('finished!')
    }
```

getCurrentFileIndex

Number **getCurrentFileIndex()**

Returns the index of the current file being transferred.

Returns

Number

Sample

```
application.output('total transferred: ' + monitor.getTotalTransferredBytes() + ' / ' + monitor.
getTotalBytesToTransfer());
    application.output('current file: ' + monitor.getCurrentTransferredFileName() + ' ( ' + monitor.
getCurrentFileIndex() + ' / ' + monitor.getTotalFilesToTransfer() + ' )');
    application.output('current bytes transferred: '+monitor.getCurrentTransferredBytes() + ' / ' + monitor.
getCurrentBytesToTransfer());
    if (monitor.isCanceled()) {
        application.output('canceled!')
    }
    if (monitor.isFinished()) {
        application.output('finished!')
    }
}
```

[getCurrentTransferredBytes](#)

Number [getCurrentTransferredBytes\(\)](#)

Returns the number of bytes already transferred for the current file.

Returns

[Number](#)

Sample

```
application.output('total transferred: ' + monitor.getTotalTransferredBytes() + ' / ' + monitor.
getTotalBytesToTransfer());
    application.output('current file: ' + monitor.getCurrentTransferredFileName() + ' ( ' + monitor.
getCurrentFileIndex() + ' / ' + monitor.getTotalFilesToTransfer() + ' )');
    application.output('current bytes transferred: '+monitor.getCurrentTransferredBytes() + ' / ' + monitor.
getCurrentBytesToTransfer());
    if (monitor.isCanceled()) {
        application.output('canceled!')
    }
    if (monitor.isFinished()) {
        application.output('finished!')
    }
}
```

[getCurrentTransferredFileName](#)

String [getCurrentTransferredFileName\(\)](#)

Returns the name of the current file being transferred.

Returns

[String](#)

Sample

```
application.output('total transferred: ' + monitor.getTotalTransferredBytes() + ' / ' + monitor.
getTotalBytesToTransfer());
    application.output('current file: ' + monitor.getCurrentTransferredFileName() + ' ( ' + monitor.
getCurrentFileIndex() + ' / ' + monitor.getTotalFilesToTransfer() + ' )');
    application.output('current bytes transferred: '+monitor.getCurrentTransferredBytes() + ' / ' + monitor.
getCurrentBytesToTransfer());
    if (monitor.isCanceled()) {
        application.output('canceled!')
    }
    if (monitor.isFinished()) {
        application.output('finished!')
    }
}
```

[getTotalBytesToTransfer](#)

Number [getTotalBytesToTransfer\(\)](#)

Returns the total bytes to transfer to or from the server (sum of all the files size)

Returns

[Number](#)

Sample

```
application.output('total transferred: ' + monitor.getTotalTransferredBytes() + ' / ' + monitor.
getTotalBytesToTransfer());
    application.output('current file: ' + monitor.getCurrentTransferredFileName() + ' ( ' + monitor.
getCurrentFileIndex() + ' / ' + monitor.getTotalFilesToTransfer() + ' )');
    application.output('current bytes transferred: '+monitor.getCurrentTransferredBytes() + ' / ' + monitor.
getCurrentBytesToTransfer());
    if (monitor.isCanceled()) {
        application.output('canceled!')
    }
    if (monitor.isFinished()) {
        application.output('finished!')
    }
}
```

`getTotalFilesToTransfer`

Number `getTotalFilesToTransfer()`

Returns the total number of files to transfer.

Returns

Number

Sample

```
application.output('total transferred: ' + monitor.getTotalTransferredBytes() + ' / ' + monitor.
getTotalBytesToTransfer());
    application.output('current file: ' + monitor.getCurrentTransferredFileName() + ' ( ' + monitor.
getCurrentFileIndex() + ' / ' + monitor.getTotalFilesToTransfer() + ' )');
    application.output('current bytes transferred: '+monitor.getCurrentTransferredBytes() + ' / ' + monitor.
getCurrentBytesToTransfer());
    if (monitor.isCanceled()) {
        application.output('canceled!')
    }
    if (monitor.isFinished()) {
        application.output('finished!')
    }
}
```

`getTotalTransferredBytes`

Number `getTotalTransferredBytes()`

Returns the total bytes already transferred (for all files)

Returns

Number

Sample

```
application.output('total transferred: ' + monitor.getTotalTransferredBytes() + ' / ' + monitor.
getTotalBytesToTransfer());
    application.output('current file: ' + monitor.getCurrentTransferredFileName() + ' ( ' + monitor.
getCurrentFileIndex() + ' / ' + monitor.getTotalFilesToTransfer() + ' )');
    application.output('current bytes transferred: '+monitor.getCurrentTransferredBytes() + ' / ' + monitor.
getCurrentBytesToTransfer());
    if (monitor.isCanceled()) {
        application.output('canceled!')
    }
    if (monitor.isFinished()) {
        application.output('finished!')
    }
}
```

`isCanceled`

Boolean `isCanceled()`

Returns true if the process was canceled.

Returns

Boolean

Sample

```
application.output('total transferred: ' + monitor.getTotalTransferredBytes() + ' / ' + monitor.
getTotalBytesToTransfer());
    application.output('current file: ' + monitor.getCurrentTransferredFileName() + ' ( ' + monitor.
getCurrentFileIndex() + ' / ' + monitor.getTotalFilesToTransfer() + ' )');
    application.output('current bytes transferred: '+monitor.getCurrentTransferredBytes() + ' / ' + monitor.
getCurrentBytesToTransfer());
    if (monitor.isCanceled()) {
        application.output('canceled!')
    }
    if (monitor.isFinished()) {
        application.output('finished!')
    }
}
```

isFinished

Boolean **isFinished()**

Returns true if the process is finished.

Returns

Boolean

Sample

```
application.output('total transferred: ' + monitor.getTotalTransferredBytes() + ' / ' + monitor.
getTotalBytesToTransfer());
    application.output('current file: ' + monitor.getCurrentTransferredFileName() + ' ( ' + monitor.
getCurrentFileIndex() + ' / ' + monitor.getTotalFilesToTransfer() + ' )');
    application.output('current bytes transferred: '+monitor.getCurrentTransferredBytes() + ' / ' + monitor.
getCurrentBytesToTransfer());
    if (monitor.isCanceled()) {
        application.output('canceled!')
    }
    if (monitor.isFinished()) {
        application.output('finished!')
    }
}
```

setProgressCallBack

JSProgressMonitor **setProgressCallBack**(progressCallbackFunction, interval, [testDelay])

Sets a method to be called repeatedly at the given interval (in seconds), the method will receive an instance of this JSProgressMonitor updated with the latest values. Can use an optional delay (for testing purpose in developer).

Parameters

progressCallbackFunction

interval

[testDelay]

Returns

JSProgressMonitor

Sample

```
// call the progressCallbackFuntion every 2 and a half seconds (with a delay of 200ms in developer):
monitor.setProgressCallBack(progressCallbackFunction, 2.5, (application.isInDeveloper() ? 200 : 0));
```