



ActionScript 3.0

www.actionscriptcheatsheet.com
info@seantheflashguy.com

Special Types

*	an property untyped
void	cannot return any value
Null	lack of a value.

Date(yrTm:Object, mo:Number, dt:Number = 1, h:Number = 0, min:Number = 0, s:Number = 0, ms:Number = 0)

Properties

date : Number
dateUTC : Number
day : Number
dayUTC : Number
fullYear : Number
fullYearUTC : Number
hours : Number
hoursUTC : Number
milliseconds : Number
millisecondsUTC : Number
minutes : Number
minutesUTC : Number
month : Number
monthUTC : Number
seconds : Number
secondsUTC : Number
time : Number
timezoneOffset : Number

Methods

getDate(): Number
getDay(): Number
getFullYear(): Number
getHours(): Number
getMilliseconds(): Number
getMinutes(): Number
getMonth(): Number
getSeconds(): Number
getTime(): Number
getTimezoneOffset(): Number
getUTCDate(): Number
getUTCDay(): Number
getUTCFullYear(): Number
getUTCHours(): Number
getUTCMilliseconds(): Number
getUTCMinutes(): Number
getUTCMonth(): Number
getUTCSeconds(): Number
hasOwnProperty(name: String): Boolean
isPrototypeOf(theClass: Object): Boolean
parse(date: String): Number
propertyIsEnumerable(name: String): Boolean
setDate(day: Number): Number
setFullYear(year: Number, month: Number, day: Number): Number
setHours(hour: Number, minute: Number, second: Number, millisecond: Number): Number
setMilliseconds(milliseconds: Number): Number
setMinutes(minute: Number, second: Number, millisecond: Number): Number
setMonth(month: Number, day: Number): Number
setPropertyIsEnumerable(name: String, isEnum: Boolean = true): void
setSeconds(second: Number, millisecond: Number): Number
setTime(millisecond: Number): Number
setUTCDate(day: Number): Number
setUTCFullYear(year: Number, month: Number, day: Number): Number
setUTCHours(hour: Number, minute: Number, second: Number, millisecond: Number): Number
setUTCMilliseconds(millisecond: Number): Number
setUTCMinutes(minute: Number, second: Number, millisecond: Number): Number
setUTCMonth(month: Number, day: Number): Number
setUTCSeconds(second: Number, millisecond: Number): Number
toString(): String
toLocaleDateString(): String
toLocaleString(): String
toLocaleTimeString(): String
toString(): String
getTimeString(): String
toUTCString(): String
valueOf(): Number

Constants

concat(... args): Array
every(callback: Function, thisObject: * = null): Boolean
filter(callback: Function, thisObject: * = null): Array
forEach(callback: Function, thisObject: * = null): void
indexOf(searchElement: *, fromIndex: int = 0): int
isPrototypeOf(theClass: Object): Boolean
join(sep: *): String
lastIndexOf(searchElement: *, fromIndex: int = 0x7fffffff): int
map(callback: Function, thisObject: * = null): Array
pop(): Object
propertyIsEnumerable(name: String): Boolean
push(... args): uint
reverse(): Array
shift(): Object
slice(startIndex: int = 0, endIndex: int = -1): Array
some(callback: Function, thisObject: * = null): Boolean
sort(... args): Array
sortOn(fieldName: Object, options: Object = null): Array
splice(startIndex: int, deleteCount: uint, ... values): Array
toLocaleString(): String
toString(): String
unshift(... args): uint

Global Constants

Constant
Infinity
-Infinity
NaN
undefined

Global Functions

Function
Array
Boolean
decodeURI
decodeURIComponent
encodeURIComponent
encodeURIComponent
escape
int
isFinite
isNaN
isXMLName
Number
Object
parseFloat
parseInt
String
trace
uint
unescape
XML
XMLList

Classes

Class
ArgumentError
arguments
Array
Boolean
Class
Date
DefinitionError
Error
EvalError
Function
int
Math
Namespace
Number
Object
 QName
RangeError
ReferenceError
RegExp
SecurityError
String
SyntaxError
TypeError
uint
URIError
VerifyError
XML
XMLList

ArgumentError(msg:String = "")
arguments

Properties

callee
length

Array(... values)

Properties

length

Methods

concat(... args): Array
every(callback: Function, thisObject: * = null): Boolean
filter(callback: Function, thisObject: * = null): Array
forEach(callback: Function, thisObject: * = null): void
indexOf(searchElement: *, fromIndex: int = 0): int
isPrototypeOf(theClass: Object): Boolean
join(sep: *): String
lastIndexOf(searchElement: *, fromIndex: int = 0x7fffffff): int
map(callback: Function, thisObject: * = null): Array
pop(): Object
propertyIsEnumerable(name: String): Boolean
push(... args): uint
reverse(): Array
shift(): Object
slice(startIndex: int = 0, endIndex: int = -1): Array
some(callback: Function, thisObject: * = null): Boolean
sort(... args): Array
sortOn(fieldName: Object, options: Object = null): Array
splice(startIndex: int, deleteCount: uint, ... values): Array
toLocaleString(): String
toString(): String
unshift(... args): uint

Boolean(exp:Object = false)

Methods

toString(): String
valueOf(): Boolean



ActionScript 3.0

www.actionscriptcheatsheet.com
info@seantheflashguy.com

DefinitionError(msg:String = "")

Error(msg:String = "", id:int = 0)

Properties

errorID : int
message : String
name : String

Methods

Error(message:String = "", id:int = 0)
getStackTrace():String
toString():String

EvalError(message:String = "")

Function

Methods

apply(thisObject:Object, argArray:Array = null):void
call(thisObject:Object, parameter1:String = null):void

int(num:Object)

Methods

toExponential(fractionDigits:uint):String
toFixed(fractionDigits:uint):String
toPrecision(precision:uint):String
toString(radix:uint):String
valueOf():int

Constants

MAX_VALUE : int = 2147483647
MIN_VALUE : int = -2147483648

Math

Methods

abs(val:Number):Number
acos(val:Number):Number
asin(val:Number):Number
atan(val:Number):Number
atan2(y:Number, x:Number):Number
ceil(val:Number):Number
cos(angleRadians:Number):Number
exp(val:Number):Number
floor(val:Number):Number
hasOwnProperty(name:String):Boolean
isPrototypeOf(theClass:Object):Boolean
log(val:Number):Number
max(val1:Number, val2:Number, ... rest):Number
min(val1:Number, val2:Number, ... rest):Number
pow(val1:Number, val2:Number):Number
propertyIsEnumerable(name:String):Boolean
random():Number
round(val:Number):Number
setPropertyIsEnumerable(n:String, isEn:Boolean = true):void
sin(angleRadians:Number):Number
sqrt(val:Number):Number
tan(angleRadians:Number):Number

Constants

E : Number = 2.71828182845905
LN10 : Number = 2.302585092994046
LN2 : Number = 0.6931471805599453
LOG10E : Number = 0.4342944819032518
LOG2E : Number = 1.442695040888963387
PI : Number = 3.141592653589793
SQRT1_2 : Number = 0.7071067811865476
SQRT2 : Number = 1.4142135623730951

Namespace(uriValue:*)

Methods

prefix : String
uri : String

Methods

Namespace(prefixValue:*, uriValue:*)
toString():String
valueOf():String

Number(num:Object)

Methods

toExponential(fractionDigits:uint):String
toFixed(fractionDigits:uint):String
toPrecision(precision:uint):String
toString(radix:Number = 10):String
valueOf():Number

Constants

MAX_VALUE : Number
MIN_VALUE : Number
NaN : Number
NEGATIVE_INFINITY : Number
POSITIVE_INFINITY : Number

Object()

Properties

constructor : Object
prototype : Object

Methods

hasOwnProperty(name:String):Boolean
isPrototypeOf(theClass:Object):Boolean
propertyIsEnumerable(name:String):Boolean
setPropertyIsEnumerable(nm:String, isEn:Boolean = true):void
toString():String
valueOf():Object

QName(qname:QName)

Properties

localName : String
uri : String

Methods

QName(uri:Namespace, localName:QName)
toString():String
valueOf():QName

RangeError(message:String = "")

ReferenceError(message:String = "")

SecurityError(message:String = "")

RegExp(re:String, flags:String)

Properties

dotall : Boolean
extended : Boolean
global : Boolean
ignoreCase : Boolean
lastIndex : Number
multiline : Boolean
source : String

Methods

exec(str:String):Object
test(str:String):Boolean

Operators

Arithmetic

+	addition
+	addition
--	decrement
/	division
++	increment
%	modulo
*	multiplication
-	subtraction

Arithmetic compound assignment

+	addition
+	addition
--	decrement
/	division
++	increment
%	modulo
*	multiplication
-	subtraction

Operators cont.

Bitwise

&	bitwise AND
<<	bitwise left shift
~	bitwise NOT
	bitwise OR
>>	bitwise right shift
>>>	bitwise unsigned right shift
^	bitwise XOR

Comparison

==	equality
>	greater than
>=	greater than or equal to
!=	inequality
<	less than
<=	less than or equal to
===	strict equality
!==	strict inequality

Logical

&&	logical AND
!	logical NOT
	logical OR

Other

[]	array access
as	as
,	comma
?:	conditional
delete	delete
.	dot
in	instanceof
is	is
::	name qualifier
new	new
{}	object initializer
()	parentheses
/	RegExp delimiter
:	type
typeof	typeof
void	void

String

+	concatenation
+=	concatenation assignment
"	string delimiter

XML

@	attribute identifier
{ }	braces (XML)
[]	brackets (XML)
+	concatenation (XMLList)
+=	concatenation assignment (XMLList)
delete	delete (XML)
..	descendant accessor
.	dot (XML)
()	parentheses (XML)
< >	XML literal tag delimiter

Bitwise compound assignment

&	bitwise AND
<<	bitwise left shift
~	bitwise NOT
	bitwise OR
>>	bitwise right shift
>>>	bitwise unsigned right shift
^	bitwise XOR

Assignment

=	assignment
---	------------

Comment

/* */	block comment delimiter
//	line comment delimiter

Top Level



String(val:String)

Properties

length : int

Methods

charAt(index: Number = 0): String
charCodeAt(index: Number = 0): Number
concat(... args): String
fromCharCode(... charCodes): String
indexOf(val: String, startIndex: Number = 0): int
lastIndexOf(val: String, startIndex: Number = 0x7FFFFFFF): int
localeCompare(other: String, ... values): int
match(pattern: *): Array
replace(pattern: *, repl: Object): String
search(pattern: *): int
slice(startIndex: Number = 0, endIndex: Number = 0x7fffffff): String
split(delimiter: *, limit: Number = 0x7fffffff): Array
substr(startIndex: Number = 0, len: Number = 0x7fffffff): String
substring(startIndex: Number = 0, endIndex: Number = 0x7fffffff): String
toLocaleLowerCase(): String
toLocaleUpperCase(): String
toLowerCase(): String
toUpperCase(): String
valueOf(): String

SyntaxError(message:String = "")

TypeError(message:String = "")

uint(num:Object)

Methods

toExponential(fractionDigits: uint): String
toFixed(fractionDigits: uint): String
toPrecision(precision: uint): String
toString(radix: uint): String
valueOf(): uint

Constants

MAX_VALUE : uint = 4294967295
MIN_VALUE : uint = 0

URIError(message:String = "")

VerifyError(message:String = "")

XMLElement(value:Object)

Methods

attribute(attributeName: *): XMLList
attributes(): XMLList
child(propertyName: Object): XMLList
children(): XMLList
comments(): XMLList
contains(value: XML): Boolean
copy(): XMLList
descendants(name: Object = *): XMLList
elements(name: Object = *): XMLList
hasComplexContent(): Boolean
hasOwnProperty(p: String): Boolean
hasSimpleContent(): Boolean
isPrototypeOf(theClass: Object): Boolean
length(): int
normalize(): XMLList
parent(): Object
processingInstructions(name: String = "*"): XMLList
propertyIsEnumerable(p: String): Boolean
setPropertyIsEnumerable(name: String, isEnum: Boolean = true): void
text(): XMLList
toString(): String
toXMLString(): StringvalueOf(): XMLList

XML(value:Object)

Properties

ignoreComments : Boolean
ignoreProcessingInstructions : Boolean
ignoreWhitespace : Boolean
prettyIndent : int
prettyPrinting : Boolean

Methods

addNamespace(ns: Object): XML
appendChild(child: Object): XML
attribute(attributeName: *): XMLList
attributes(): XMLList
child(propertyName: Object): XMLList
childIndex(): int
children(): XMLList
comments(): XMLList
contains(value: XML): Boolean
copy(): XML
defaultSettings(): Object
descendants(name: Object = *): XMLList
elements(name: Object = *): XMLList
hasComplexContent(): Boolean
hasOwnProperty(p: String): Boolean
hasSimpleContent(): Boolean
inScopeNamespaces(): Array
insertChildAfter(child1: Object, child2: Object): *
insertChildBefore(child1: Object, child2: Object): *
length(): int
localName(): Object
name(): Object
namespace(prefix: String = null): *
namespaceDeclarations(): Array
nodeKind(): String
normalize(): XML
parent(): *
prependChild(value: Object): XML
processingInstructions(name: String = "*"): XMLList
propertyIsEnumerable(p: String): Boolean
removeNamespace(ns: Namespace): XML
replace(propertyName: Object, value: XML): XML
setChildren(value: Object): XML
setLocalName(name: String): void
setName(name: String): void
setNamespace(ns: Namespace): void
setSettings(... rest): void
settings(): Object
text(): XMLList
toString(): String
toXMLString(): String
valueOf(): XML

Statements, Keywords & Directives

Statement summary

break
case
continue
default
do..while
else
for
for..in
for each..in
if
label
return
super
switch
throw
try..catch..finally
while
with

Attribute keyword summary

dynamic
final
internal
native
override
private
protected
public
static

Definition keyword summary

... (rest) parameter
class
const
extends
function
get
implements
interface
namespace
package
set
var

Directive summary

default xml namespace
import
include
use namespace

Namespace summary

AS3
flash_proxy
object_proxy

Primary expression keyword

false
null
this
true