Cheatography

Closure JSDoc Cheat Sheet

by killermonkeys via cheatography.com/2120/cs/611/

JSDoc Tags for Functions

@constructor	Function *is a constructor (can* new*)*
@deprecated	Function *is deprecated*
@extends {Type}	Function *inherits* Type
@implements {Type}	Function *implements* Type *(with* @constructor*)*
@inheritDoc	Function *has same JSDoc as superclass*
@interface	Function *is interface (no* new*)*
@nosideeffects	*Can be removed if return value not used*
@override	Function *overrides superclass*
@param {Type} varname Description	Function *takes* varname *of* Type
@private	Function *is private (same file or static/instance members)*
@protected	Function *is protected (same file or static/instance of subclasses)*
@return {Type} Description	Function *returns* Type
@this {Type}	*In* Function, this *is* Type

JSDoc Tags for Properties

@const	Property *is constant*
@define	Property *can be overridden by compiler*
@deprecated	Property *is deprecated*
@enum {Type}	Property *is an enum of* Type (default number)
@expose	Property *not optimized by compiler*
@lends {objectName}	Keys of object are same as property of other object (see: http://code.google.com/p/jsdoc-toolkit/wiki/TagLends)
@private	Property is private
@protected	Property is protected
@type {Type}	Property is {Type}

JSDoc Type Definitions

{boolean}	True
{number}	1
{string}	'monkey'
{Object}	{}
{Array}	[]
{Window}	*defined type* Window
{goog.ui.Menu}	*defined type* goog.ui.Menu
{Array.<string>}	['a','b','c']
{Object.<string, number>}	{'a':1, 'b':2}
{(number|boolean)}	1 *or* True
{{myNum: number, myObject}}	Record *with property* myNum {number} and myObject {Object}

JSDoc Type Definitions (cont)

{Array.<{length}>}	Array *of* {Objects} *with property* length
{?number}	{number} *or* null
{!Object}	{Object} *but never* null
{function(string, boolean)}	Function *with params and unknown return value*
{function(): number}	Function *returning* number
{function(this:goog.ui.Menu, string)}	Function *where* this *is* goog.ui.Menu
{function(new:goog.ui.Menu, string)}	Function *takes* string, *creates new* goog.ui.Menu
{function(string, ... [number])}	Function *takes* string *then optional* number *s*
@param {...number} var_args	*Variable number of parameters of type* number
@param {number=} opt_argument	*Optional parameter of type* number
{function(?string=, number=)}	Function *with optional parameters*
{*}	Variable can take any type
{?}	Variable can take any type and don't type check

By **killermonkeys**

cheatography.com/killermonkeys/

Published 9th October, 2012.
Last updated 9th October, 2012.

JSDoc Example

```
/**
* Creates an instance of Circle.
*
* @constructor
* @this {Circle}
* @param {number} r The desired radius of the circle.
*/
function Circle(r) {
/* @private */ this.radius = r;
/* @private */ this.circumference = 2*Math.PI* r;
}
/**
* Creates a new Circle from a diameter.
*
* @param {number} d The desired diameter of the circle.
* @return {Circle} The new Circle object.
*/
Circle.fromDiameter = function (d) {
return new Circle(d / 2);
};
/**
* Calculates the circumference of the Circle.
*
* @deprecated
* @this {Circle}
* @return {number} The circumference of the circle.
*/
```

JSDoc Example (cont)

```
Circle.prototype.calculateCircumference =
function () {
return 2*Math.PI* this.radius;
};
/**
* Returns the pre-computed circumference of the Circle.
*
* @this {Circle}
* @return {number} The circumference of the circle.
*/
Circle.prototype.getCircumference = function ()
{
return this.circumference;
};
/**
* Find a String representation of the Circle.
*
* @override
* @this {Circle}
* @return {string} Human-readable representation of this Circle.
*/
Circle.prototype.toString = function () {
return "A Circle object with radius of " +
this.radius + ".";
};
```

By **killermonkeys**

cheatography.com/killermonkeys/

Published 9th October, 2012.
Last updated 9th October, 2012.