**Contents**

[Gooi.Assert 2](#_Toc342642107)

[Gooi.Core 3](#_Toc342642108)

[Gooi.Core.Socket 6](#_Toc342642109)

[Gooi.Helper.String 7](#_Toc342642110)

[(Reference) Lift Notes 8](#_Toc342642111)

# Gooi.Assert

/\*jshint asi:true, supernew:true \*/

var Gooi\_Globals\_Assert\_ignore = false

var Gooi\_Globals\_Assert\_WARN = false

var Gooi = Gooi || {};

Gooi.assert = function () {

this.AssertException = function () {

var ExceptionBase = {}

ExceptionBase.message = arguments[1]

ExceptionBase.valid = arguments[0]

ExceptionBase.args = Array.prototype.slice.call(arguments)

ExceptionBase.toString = function () {

return 'AssertException: ' + this.args.slice(1).join(' : ')

}

return ExceptionBase;

}

if ((!arguments[0]) && Gooi\_Globals\_Assert\_ignore !== true) {

if (!Gooi\_Globals\_Assert\_WARN) {

throw this.AssertException.apply(this, arguments)

} else {

var ex = this.AssertException.apply(this, arguments)

console.warn(ex, ex.message)

}

}

}

Gooi.Assert = (function (base) {

var \_ignore = function () { return Gooi\_Globals\_Assert\_ignore }

base.isArray = function (val1, message) {

if (!base.ignore) {

return Gooi.assert(Array.isArray(val1), message || "Value is not an Array")

}

};

base.isDate = function (val1, message) {

if (!base.ignore) {

return Gooi.assert(val1 instanceof Date && !isNaN(val1.getTime()), message || "Value is not a date")

}

};

base.areEqual = function (val1, val2, message) {

if (!base.ignore)

return Gooi.assert(val1 === val2, message || "Values are not equal")

};

base.isInstanceOf = function (val, type, message) {

if (!base.ignore)

return Gooi.assert(val instanceof type, message || "Value is not an instance of " + type.totoString)

};

base.isNumber = function (val1, message) {

if (!base.ignore)

return Gooi.assert(typeof val1 === 'number', message || "Value is not a number")

};

base.isString = function (val1, message) {

if (!base.ignore)

return Gooi.assert(typeof val1 === 'string', message || "Value is not a string")

};

//Set Global Assert Ignore

Object.defineProperty(base, 'ignore', {

get: function () { return Gooi\_Globals\_Assert\_ignore }

, set: function (value) { Gooi\_Globals\_Assert\_ignore = value }

});

//Set Global Assert Warn Only

Object.defineProperty(base, 'warn', {

get: function () { return Gooi\_Globals\_Assert\_WARN }

, set: function (value) { Gooi\_Globals\_Assert\_WARN = value }

});

base.init = function () {

if (Gooi\_Globals\_Log)

console.log('Assertion Lib Init Event Fired')

}

return base;

} (Gooi.Assert || {}));

# Gooi.Core

/\*jshint asi:true, supernew:true \*/

function gooi() {

var gooi\_base = {

Global: window

, onReady[[1]](#endnote-1): function (fn) {

if (Gooi\_Globals\_Log)

console.log('Document Ready Fired')

var timer, ready = false, setup = false, stack = [];

function StateChange(e) {

e = e || window.event;

if (window.Gooi\_Globals\_Loader\_Complete) {

if (e && e.type && /DOMContentLoaded|load/.test(e.type)) {

Ready()

} else if (window.document.readyState) {

if (/loaded|complete/.test(window.document.readyState)) {

Ready()

} else if (window.document.documentElement.doScroll) {

try {

ready || window.document.documentElement.doScroll('left')

} catch (e) {

return

}

Ready() //If no error was thrown, the DOM must be ready

}

}

}

};

function Ready() {

if (!ready) {

ready = true;

// Call the stack of onload functions in given context or window object

for (var i = 0, len = stack.length; i < len; i++) {

stack[i][0].call(stack[i][1]);

}

// Clean up after the DOM is ready

if (window.document.removeEventListener)

window.document.removeEventListener("DOMContentLoaded", StateChange, false)

clearInterval(timer);

//Null the timer and event handlers to release memory

window.document.onreadystatechange = window.onload = timer = null;

}

};

/\*if( ready ){ //I am not a big fan of this block. . .

//If the DOM is ready, call the function and return

fn.call( window );

return;

}\*/

if (!setup) {

debugger;

setup = true

if (window.document.addEventListener)

window.document.addEventListener("DOMContentLoaded", StateChange, false)

timer = setInterval(StateChange, 5)

window.document.onreadystatechange = window.onload = StateChange

}

stack.push([fn]);

}

, Core: function () {

var gooi\_core\_base = { Settings: {} }

gooi\_core\_base.Extend = function (destination, source) {

for (var member in source) {

//debugger;

destination[member] = source[member]

}

return destination

};

/\*Needs to allow setting of defaulted values here.\*/

//Global.Gooi\_Globals\_Site = '/gdirlam/gooi/workspace'

gooi\_core\_base.init = function () {

gooi\_core\_base.Extend(gooi\_core\_base.Settings, Array.prototype.slice.call(arguments)[0])

for (var member in gooi\_core\_base.Settings) {

window['Gooi\_Globals\_' + member] = gooi\_core\_base.Settings[member]

}

};

gooi\_core\_base.init(Array.prototype.slice.call(arguments)[0])

gooi\_core\_base.Bind = function (caller, object) {

return function () {

return caller.apply(object, [object])

};

};

gooi\_core\_base.Loaded = [];

gooi\_core\_base.Loader = function () {

var gooi\_core\_loader\_base = {}

var Global = window //need better way to reference the parent

//debugger;

gooi\_core\_loader\_base.Asset = function (name, location) {

var base = {

Name: name

, Location: location

, Url: function () {

//debugger;

return Global.Gooi\_Globals\_Site + location + ((!/(\.js)$/.test(location)) ? '.js' : '')

}

};

return base

}

Global.Gooi\_Globals\_Assets = [];

Global.Gooi\_Globals\_Assets['Gooi.Core'] = new gooi\_core\_loader\_base.Asset('Gooi.Core', '/GooiCore/GooiCore.js')

Global.Gooi\_Globals\_Assets['Gooi.Assert'] = new gooi\_core\_loader\_base.Asset('Gooi.Assert', '/GooiAssert/GooiAssert.js')

Global.Gooi\_Globals\_Assets['Gooi.Core.Socket'] = new gooi\_core\_loader\_base.Asset('Gooi.Core.Socket', '/GooiCore/GooiCoreSocket.js')

Global.Gooi\_Globals\_Assets['Gooi.Helper.String'] = new gooi\_core\_loader\_base.Asset('Gooi.Helper.String', '/GooiHelper/GooiHelperString.js')

Global.Gooi\_Globals\_Loader\_Complete = false;

Global.Gooi\_Globals\_Loader\_Complete = false;

gooi\_core\_loader\_base.Queue = []

gooi\_core\_loader\_base.enqueue = function (asset) {

// debugger;

gooi\_core\_loader\_base.Queue[gooi\_core\_loader\_base.Queue.length] = asset

//Gooi.Core.Loaded[ Gooi.Core.Loaded.length ] = asset

}

gooi\_core\_loader\_base.Load = function () {

// debugger;

if (Gooi\_Globals\_Log)

console.log('Script Loader Loading')

do {

var asset = gooi\_core\_loader\_base.Queue[0]

gooi\_core\_loader\_base.LoadScript(asset)

gooi\_core\_loader\_base.Queue.splice(0, 1)

//debugger;

if (Gooi\_Globals\_Log)

console.log('Script Loader Loaded', asset)

} while (gooi\_core\_loader\_base.Queue.length > 0)

}

gooi\_core\_loader\_base.Use = function (library) {

gooi\_core\_loader\_base.Use.Success = function (which) {

//debugger;

if (Gooi\_Globals\_Log)

console.log('Script Loader Loaded Callback Fired', which.target.asset)

try {

//debugger;

var Fn = Function, ret = new Fn(which.target.asset.Name + '.init()')()

} catch (e) {

if (Gooi\_Globals\_Log)

console.log('Script Loader Library, init function unavailable', which.target.asset)

}

if (gooi\_core\_loader\_base.Queue.length === 0) {

Global.Gooi\_Globals\_Loader\_Complete = true

if (Gooi\_Globals\_Log)

console.log('Script Loader Complete')

}

}

// debugger;

gooi\_core\_loader\_base.enqueue(Global.Gooi\_Globals\_Assets[library])

return gooi\_core\_loader\_base

};

gooi\_core\_loader\_base.LoadScript = function (asset) {

// debugger;

var script = document.createElement('script')

var ieLoadBugFix [[2]](#endnote-2)= function (scriptElement, callback) {

if (scriptElement.readyState == 'loaded' || scriptElement.readyState == 'complete') {

scriptElement.asset = asset//this might introduce a bug here, with asset passing in.

callback();

} else {

setTimeout(function () { ieLoadBugFix(scriptElement, callback); }, 100);

}

}

script.src = asset.Url()

//script.onload = Gooi.Core.Loader.Requires.Success //does not work in ie 7

if (typeof script.addEventListener !== "undefined") {

// debugger;

script.asset = asset

//debugger;

script.addEventListener("load", this.Use.Success, false)

} else {

script.onreadystatechange = function () {

script.onreadystatechange = null;

ieLoadBugFix(script, this.Use.Success);

}

}

script.type = 'text/javascript'

document.head.appendChild(script)

//debugger;

if (Gooi\_Globals\_Log)

console.log('Script Loader Appended Script to Document')

};

//gooi\_core\_loader\_base.Use = function(){}

return gooi\_core\_loader\_base;

}

return gooi\_core\_base

}

}

return gooi\_base

}

/\* Establish Gooi Namespace based on gooi class functionality \*/

var Gooi = new gooi()

var Goo = Gooi.onReady

# Gooi.Core.Socket

/\*jshint asi:true, supernew:true \*/

//var Gooi = ( Gooi || {} )

//Gooi.Core = ( Gooi.Core || {} )

/\*requires Gooi.Core\*/

Gooi.Core.Socket = (function (base) {

base.Remote = function (url, func) {

base.Remote = (base.Remote || {}) //callback function here.

base.Remote.url = url

base.Remote.Response = {}

base.Remote.Response.Type = 'jsonp'

base.Remote.Success = func //Gooi.Core.Extend( , { responseType: 'jsonp' } )

var \_script = document.createElement('script')

\_script.src = base.Remote.url + '?callback=Gooi.Core.Socket.Remote.Success'

\_script.type = 'text/javascript'

document.body.appendChild(\_script)

return base.Remote

}

base.Web = function (url, func, settings) {

base.Web.Response = { Text: '', Type: 'json', ContentType: 'text/plain', Data: null }

base.Web.Url = url

base.Web.PostBody = (arguments[2] || '')

base.Web.Callback = func

base.Web.stateChange = function (object) {

if (base.Web.Request.readyState == 4) {

var response = base.Web.Response

response.Text = base.Web.Request.responseText.toString()

response.ContentType = base.Web.Request.getResponseHeader('content-type')

if (response.ContentType.indexOf('text') > -1)

response.Type = 'text'

if (response.ContentType.indexOf('html') > -1)

response.Type = 'html'

if (response.ContentType.indexOf('xml') > -1)

response.Type = 'xml'

if (response.ContentType.indexOf('json') > -1)

response.Type = 'json'

if (response.Type === 'json') {

response.Data = (new Function("return " + base.Web.Response.Text))()

} else {

response.Data = response.Text

}

base.Web.Callback(response.Data)

}

};

base.Web.Request = (function () {

if (window.ActiveXObject)

return new ActiveXObject('Microsoft.XMLHTTP')

else if (window.XMLHttpRequest)

return new XMLHttpRequest()

return false

})(); //iefe lets find out if ie/moz right away

base.Web.Request.Send = function () {

var request = base.Web.Request

request.onreadystatechange = Gooi.Core.Bind(base.Web.stateChange, base.Web)

if (base.Web.PostBody !== "") {

request.open("POST", base.Web.Url, true)

request.setRequestHeader('X-Requested-With', 'XMLHttpRequest')

request.setRequestHeader('Content-type', 'application/x-www-form-urlencoded')

request.setRequestHeader('Connection', 'close')

} else {

request.open("GET", base.Web.Url, true)

}

request.send(base.Web.PostBody)

}

//base.Web.Request.Send();

base.init = function () {

base.Web.Request()

}

}

Gooi.Core.Loaded['Gooi.Core.Socket.Web'] = base.Web

return base;

})(Gooi.Core.Socket || {});

# Gooi.Helper.String

/\*jshint asi:true, supernew:true \*/

var Gooi\_Globals\_Prototype = Gooi\_Globals\_Prototype || false

var Gooi = Gooi || {};

Gooi.Helper = (function (base, Global) {

Object.defineProperty(base, 'ignore', {

get: function () { return Global.Gooi\_Globals\_Prototype }

, set: function (value) { Global.Gooi\_Globals\_Prototype = value }

});

return base

})((Gooi.Helper || {}), window)

Gooi.Helper.String = {

format: function () {

var txt = this;

for (var i = 0; i < arguments.length; i++) {

var exp = new RegExp('\\{' + (i) + '\\}', 'gm')

txt = txt.replace(exp, arguments[i])

}

return txt

}

, inlineformat: function () {

for (var i = 1; i < arguments.length; i++) {

var exp = new RegExp('\\{' + (i - 1) + '\\}', 'gm')

arguments[0] = arguments[0].replace(exp, arguments[i])

}

return arguments[0]

}

, write: function () {

if (arguments.length === 0) {

var txt = this

document.body.innerHTML += '<br />' + txt + '<br />'

//document.write( this + '<br />' )

return ''

}

var txt = ''

for (var i = 0; i < arguments.length; i++)

txt += arguments[i] + '<br />'

//document.write(txt)

document.body.innerHTML += '<br />' + txt + '<br />'

return ''

}

, init: function () {

//debugger;

if (Gooi\_Globals\_Log)

console.log('String Helper Init Event Fired')

if (Gooi\_Globals\_Prototype) {

if (!String.prototype.format)

String.prototype.format = Gooi.Helper.String.format

if (!String.format)

String.format = Gooi.Helper.String.inlineformat

if (!String.write)

String.write = Gooi.Helper.String.write

if (!String.prototype.write)

String.prototype.write = Gooi.Helper.String.write

}

}

};

# (Reference) Lift Notes

**General** :

//developer.mozilla.org/en-US/docs/JavaScript/Reference/Global\_Objects/Array

1. onReady Lifted from https://github.com/ryanmorr/ondomready [↑](#endnote-ref-1)
2. IE Load Fix http://stackoverflow.com/questions/6568890/javascript-dynamic-script-loading-ie-problems [↑](#endnote-ref-2)