# bstr CWSTR Class »



## **Description**

Returns the contents of the CWSTR as a BSTR.

# **Syntax**

FUNCTION bstr () AS AFX\_BSTR

## **Parameter**

This method has no parameters.

## **Return value**

The current contents of the CWSTR as a BSTR.

## **Include file**

CWStr.inc

Created with the Personal Edition of HelpNDoc: <u>Easily create PDF Help documents</u>

# cbstr CWSTR Class »



## **Description**

Returns the contents of the CWSTR as a CBSTR.

## **Syntax**

FUNCTION cbstr () AS CBStr\_

#### **Parameter**

This method has no parameters.

## **Return value**

The current contents of the CWSTR as a CBSTR.

#### Remarks

Useful to pass a CWSTR to a function that expects a BYVAL IN BSTR parameter.

### Include file

CWStr.inc

Created with the Personal Edition of HelpNDoc: <u>Create iPhone web-based documentation</u>

# sptr CWSTR Class >>



# **Description**

Returns the address of the string data. Same as \*.

# **Syntax**

FUNCTION sptr () AS WSTRING PTR

# **Include file**

CWStr.inc

Created with the Personal Edition of HelpNDoc: <u>Easily create iPhone</u> <u>documentation</u>

# vptr CWSTR Class »



# **Description**

Returns the address of the **CWSTR** buffer.

# **Syntax**

FUNCTION vptr () AS WSTRING PTR

# **Include file**

CWStr.inc

Created with the Personal Edition of HelpNDoc: <u>Generate EPub eBooks</u> with ease



Returns the string data as a new unicode string allocated with CoTaskMemAlloc.

## **Syntax**

FUNCTION wchar () AS WSTRING PTR

#### Remarks

The new string must be freed with CoTaskMemFree.

Useful when we need to pass a pointer to a null terminated wide string to a function or method that will release it. If we pass a WSTRING it will GPF. If the length of the input string is 0, CoTaskMemAlloc allocates a zero-length item and returns a valid pointer to that item. If there is insufficient memory available, CoTaskMemAlloc returns NULL.

# **Usage example**

DIM cws AS CWSTR = "Test string"
DIM pwchar AS WSTRING PTR = cws.wchar
PRINT \*pwchar
CoTaskMemFree pwchar

#### Include file

CWStr.inc

Created with the Personal Edition of HelpNDoc: <u>Free CHM Help</u> <u>documentation generator</u>

# WSTR Class »



# **Description**

Returns the string data. Same as \*\*.

# **Syntax**

FUNCTION wstr () BYREF AS WSTRING

# **Include file**

CWStr.inc

Created with the Personal Edition of HelpNDoc: <u>Easily create EBooks</u>

# **bptr**CBSTR Class »



# **Description**

Returns the underlying BSTR pointer.

# **Syntax**

FUNCTION bptr () BYREF AS AFX\_BSTR

# **Include file**

CWStr.inc

Created with the Personal Edition of HelpNDoc: <u>Free CHM Help documentation generator</u>

# Sptr CBSTR Class >>



## **Description**

Returns the address of the string data. Same as \*\*.

# **Syntax**

FUNCTION sptr () AS WSTRING PTR

# **Include file**

CWStr.inc

Created with the Personal Edition of HelpNDoc: <u>Create help files for the</u>
<a href="Qt Help Framework">Qt Help Framework</a>

# vptr CBSTR Class »



## **Description**

Frees the underlying BSTR and returns the BSTR pointer.

## **Syntax**

FUNCTION vptr () AS WSTRING PTR

#### Remarks

To pass the underlying BSTR to an OUT BYVAL BSTR PTR parameter. If we pass a CBSTR to a function with an OUT BSTR parameter without first freeing it we will have a memory leak.

### **Include file**

CWStr.inc

Created with the Personal Edition of HelpNDoc: <u>Create HTML Help</u>, <u>DOC</u>, <u>PDF</u> and print manuals from 1 single source



Returns the string data as a new unicode string allocated with CoTaskMemAlloc.

## **Syntax**

FUNCTION wchar () AS WSTRING PTR

#### Remarks

The new string must be freed with CoTaskMemFree.

Useful when we need to pass a pointer to a null terminated wide string to a function or method that will release it. If we pass a WSTRING it will GPF. If the length of the input string is 0, CoTaskMemAlloc allocates a zero-length item and returns a valid pointer to that item. If there is insufficient memory available, CoTaskMemAlloc returns NULL.

# **Usage example**

DIM cbs AS CBSTR = "Test string"
DIM pwchar AS WSTRING PTR = cbs.wchar
PRINT \*pwchar
CoTaskMemFree pwchar

#### Include file

CWStr.inc

Created with the Personal Edition of HelpNDoc: <u>Free PDF documentation</u> <u>generator</u>

# WStr CBSTR Class »



# **Description**

Returns the string data. Same as \*\*.

# **Syntax**

FUNCTION wstr () BYREF AS WSTRING

# **Include file**

CWStr.inc

Created with the Personal Edition of HelpNDoc: <u>Free iPhone</u> <u>documentation generator</u>

# bstr

CVAR Class >>



## **Description**

Extracts the content of the underlying variant and returns it as a CBSTR. Same as ToBStr.

## **Syntax**

```
FUNCTION CVar.bstr () AS CBSTR
```

#### **Parameter**

This method has no parameters.

### Return value

The contents of the variant as a CBSTR.

# Example

```
DIM cv AS CVAR = "Test string"
DIM cbs AS CBSTR = cv.bstr
```

#### Include file

CVar.inc

Created with the Personal Edition of HelpNDoc: <u>Single source CHM</u>, <u>PDF, DOC and HTML Help creation</u>



Returns the address of the underlying variant. Same as vptr but without clearing the variant.

## **Syntax**

```
FUNCTION CVar.sptr () AS VARIANT PTR
```

#### **Parameter**

This method has no parameters.

#### Return value

The address of the underlying variant. Can be used to pass the variant to a IN BYVAL VARIANT PTR parameter.

# **Example**

```
SUB FOO (BYVAL v AS VARIANT PTR)
PRINT AfxVarToStr(v)
END SUB
```

Using the pointer syntax:

```
DIM pcv AS CVAR PTR = NEW CVAR("Test string")
Foo pcv->sptr
Delete pcv
```

Using the normal syntax:

```
DIM cv AS CVAR = "Test string"
Foo cv.sptr
```

But with the normal syntax you can use the \* operator instead:

```
DIM cv AS CVAR = "Test string"
Foo *cv
```

or simply

```
DIM cv AS CVAR = "Test string"
Foo cv
```

# **Include file**

CVar.inc

Created with the Personal Edition of HelpNDoc: <u>Easy CHM and</u> <u>documentation editor</u>



Clears the CVAR and returns the address of the underlying variant. Can be used to pass the variant to an OUT BYVAL VARIANT PTR parameter. If we pass a CVAR to a function with an OUT variant parameter without first clearing the contents of the CVAR, we may have a memory leak.

## **Syntax**

```
FUNCTION vptr () AS VARIANT PTR
```

#### **Parameter**

This method has no parameters.

#### Return value

The address of the underlying variant.

# **Example**

```
SUB FOO (BYVAL v AS VARIANT PTR)
v->vt = VT_I4
v->lVal = 12345
END SUB

DIM cv AS CVAR = "Test string"
FOO cv.vptr
PRINT cv
```

Otherwise, you need to clear the underlying variant before passing the CVAR.

```
DIM cv AS CVAR = "Test string"
cv.Clear
Foo *cv
```

# **PRINT** cv

# **Include file**

CVar.inc

Created with the Personal Edition of HelpNDoc: <u>Easily create EBooks</u>

WStr CVAR Class »



## **Description**

Extracts the content of the underlying variant and returns it as a CWSTR. Same as ToStr.

## **Syntax**

```
FUNCTION wstr () AS CWSTR
```

#### **Parameter**

This method has no parameters.

### **Return value**

The contents of the variant as a CWSTR.

# Example

```
DIM cv AS CVAR = "Test string"
DIM cws AS CWSTR = cv.wstr
```

### Include file

CVar.inc

Created with the Personal Edition of HelpNDoc: <u>Full-featured EPub</u> <u>generator</u>

# vptr

CDispInvoke Class >>



## **Description**

Clears the contents of the class and returns the address of the underlying IDispatch pointer.

## **Syntax**

FUNCTION vptr () AS IDIspatch PTR PTR

#### **Parameters**

#### cdi

[in] An instance of the CDispInvoke class.

#### Remarks

To pass the class to an OUT BYVAL IDispatch PTR parameter.

If we pass a CDispInvoke to a function with an OUT IDIspatch parameter without first clearing the contents of the class, we may have a memory leak.

### **Include file**

CDispInvoke.inc

Created with the Personal Edition of HelpNDoc: <u>Generate Kindle eBooks</u> <u>with ease</u>