# Body Object (JavaScript API for Word)

Word 2016, Word for iPad, Word for Mac

Represents the body of a document or a section.

## Properties

| **Property** | **Type** | **Description** | **Req. Set** |
| --- | --- | --- | --- |
| style | string | Gets or sets the style used for the body. This is the name of the pre-installed or custom style. | 1.1 |  |
| text | string | Gets the text of the body. Use the insertText method to insert text. Read-only. | 1.1 |  |
| type | string | Gets the type of the body. The type can be 'MainDoc', 'Section', 'Header', 'Footer', or 'TableCell'. Read-only. Possible values are: Unknown, MainDoc, Section, Header, Footer, TableCell. | 1.3 |  |

See property access [*examples.*](#property-access-examples)

## Relationships

| **Relationship** | **Type** | **Description** | **Req. Set** |
| --- | --- | --- | --- |
| contentControls | [ContentControlCollection](file:///C:\Users\chbigham\Documents\contentcontrolcollection.md) | Gets the collection of rich text content control objects in the body. Read-only. | 1.1 |  |
| font | [Font](file:///C:\Users\chbigham\Documents\font.md) | Gets the text format of the body. Use this to get and set font name, size, color and other properties. Read-only. | 1.1 |  |
| inlinePictures | [InlinePictureCollection](file:///C:\Users\chbigham\Documents\inlinepicturecollection.md) | Gets the collection of inlinePicture objects in the body. The collection does not include floating images. Read-only. | 1.1 |  |
| lists | [ListCollection](file:///C:\Users\chbigham\Documents\listcollection.md) | Gets the collection of list objects in the body. Read-only. | 1.3 |  |
| paragraphs | [ParagraphCollection](file:///C:\Users\chbigham\Documents\paragraphcollection.md) | Gets the collection of paragraph objects in the body. Read-only. | 1.1 |  |
| parentBody | [Body](file:///C:\Users\chbigham\Documents\body.md) | Gets the parent body of the body. For example, a table cell body's parent body could be a header. Read-only. | 1.3 |  |
| parentContentControl | [ContentControl](file:///C:\Users\chbigham\Documents\contentcontrol.md) | Gets the content control that contains the body. Returns null if there isn't a parent content control. Read-only. | 1.1 |  |
| tables | [TableCollection](file:///C:\Users\chbigham\Documents\tablecollection.md) | Gets the collection of table objects in the body. Read-only. | 1.3 |  |

## Methods

| **Method** | **Return Type** | **Description** | **Req. Set** |
| --- | --- | --- | --- |
| [clear()](#clear) | void | Clears the contents of the body object. The user can perform the undo operation on the cleared content. | 1.1 |
| [getHtml()](#gethtml) | string | Gets the HTML representation of the body object. | 1.1 |
| [getOoxml()](#getooxml) | string | Gets the OOXML (Office Open XML) representation of the body object. | 1.1 |
| [getRange(rangeLocation: string)](#getrangerangelocation-string) | [Range](file:///C:\Users\chbigham\Documents\range.md) | Gets the whole body, or the starting or ending point of the body, as a range. | 1.3 |
| [insertBreak(breakType: string, insertLocation: string)](#insertbreakbreaktype-string-insertlocat) | void | Inserts a break at the specified location in the main document. The insertLocation value can be 'Start' or 'End'. | 1.1 |
| [insertContentControl()](#insertcontentcontrol) | [ContentControl](file:///C:\Users\chbigham\Documents\contentcontrol.md) | Wraps the body object with a Rich Text content control. | 1.1 |
| [insertFileFromBase64(base64File: string, insertLocation: string)](#insertfilefrombase64base64file-string-i) | [Range](file:///C:\Users\chbigham\Documents\range.md) | Inserts a document into the body at the specified location. The insertLocation value can be 'Replace', 'Start' or 'End'. | 1.1 |
| [insertHtml(html: string, insertLocation: string)](#inserthtmlhtml-string-insertlocation-st) | [Range](file:///C:\Users\chbigham\Documents\range.md) | Inserts HTML at the specified location. The insertLocation value can be 'Replace', 'Start' or 'End'. | 1.1 |
| [insertInlinePictureFromBase64(base64EncodedImage: string, insertLocation: string)](#insertinlinepicturefrombase64base64enco) | [InlinePicture](file:///C:\Users\chbigham\Documents\inlinepicture.md) | Inserts a picture into the body at the specified location. The insertLocation value can be 'Start' or 'End'. | 1.2 |
| [insertOoxml(ooxml: string, insertLocation: string)](#insertooxmlooxml-string-insertlocation-) | [Range](file:///C:\Users\chbigham\Documents\range.md) | Inserts OOXML at the specified location. The insertLocation value can be 'Replace', 'Start' or 'End'. | 1.1 |
| [insertParagraph(paragraphText: string, insertLocation: string)](#insertparagraphparagraphtext-string-ins) | [Paragraph](file:///C:\Users\chbigham\Documents\paragraph.md) | Inserts a paragraph at the specified location. The insertLocation value can be 'Start' or 'End'. | 1.1 |
| [insertTable(rowCount: number, columnCount: number, insertLocation: string, values: string[][])](#inserttablerowcount-number-columncount-) | [Table](file:///C:\Users\chbigham\Documents\table.md) | Inserts a table with the specified number of rows and columns. The insertLocation value can be 'Start' or 'End'. | 1.3 |
| [insertText(text: string, insertLocation: string)](#inserttexttext-string-insertlocation-st) | [Range](file:///C:\Users\chbigham\Documents\range.md) | Inserts text into the body at the specified location. The insertLocation value can be 'Replace', 'Start' or 'End'. | 1.1 |
| [load(param: object)](#loadparam-object) | void | Fills the proxy object created in JavaScript layer with property and object values specified in the parameter. | 1.1 |
| [search(searchText: string, searchOptions: ParamTypeStrings.SearchOptions)](#searchsearchtext-string-searchoptions-p) | [SearchResultCollection](file:///C:\Users\chbigham\Documents\searchresultcollection.md) | Performs a search with the specified searchOptions on the scope of the body object. The search results are a collection of range objects. | 1.1 |
| [select(selectionMode: string)](#selectselectionmode-string) | void | Selects the body and navigates the Word UI to it. | 1.1 |

## Method Details

### clear()

Clears the contents of the body object. The user can perform the undo operation on the cleared content.

#### Syntax

bodyObject.clear();

#### Parameters

None

#### Returns

void

#### Examples

// Run a batch operation against the Word object model.

Word.run(function (context) {

// Create a proxy object for the document body.

var body = context.document.body;

// Queue a commmand to clear the contents of the body.

body.clear();

// Synchronize the document state by executing the queued commands,

// and return a promise to indicate task completion.

return context.sync().then(function () {

console.log('Cleared the body contents.');

});

})

.catch(function (error) {

console.log("Error: " + JSON.stringify(error));

if (error instanceof OfficeExtension.Error) {

console.log("Debug info: " + JSON.stringify(error.debugInfo));

}

});

The [Silly stories](https://aka.ms/sillystorywordaddin) add-in sample shows how the **clear** method can be used to clear the contents of a document.

### getHtml()

Gets the HTML representation of the body object.

#### Syntax

bodyObject.getHtml();

#### Parameters

None

#### Returns

string

#### Examples

// Run a batch operation against the Word object model.

Word.run(function (context) {

// Create a proxy object for the document body.

var body = context.document.body;

// Queue a commmand to get the HTML contents of the body.

var bodyHTML = body.getHtml();

// Synchronize the document state by executing the queued commands,

// and return a promise to indicate task completion.

return context.sync().then(function () {

console.log("Body HTML contents: " + bodyHTML.value);

});

})

.catch(function (error) {

console.log("Error: " + JSON.stringify(error));

if (error instanceof OfficeExtension.Error) {

console.log("Debug info: " + JSON.stringify(error.debugInfo));

}

});

### getOoxml()

Gets the OOXML (Office Open XML) representation of the body object.

#### Syntax

bodyObject.getOoxml();

#### Parameters

None

#### Returns

string

#### Examples

// Run a batch operation against the Word object model.

Word.run(function (context) {

// Create a proxy object for the document body.

var body = context.document.body;

// Queue a commmand to get the OOXML contents of the body.

var bodyOOXML = body.getOoxml();

// Synchronize the document state by executing the queued commands,

// and return a promise to indicate task completion.

return context.sync().then(function () {

console.log("Body OOXML contents: " + bodyOOXML.value);

});

})

.catch(function (error) {

console.log("Error: " + JSON.stringify(error));

if (error instanceof OfficeExtension.Error) {

console.log("Debug info: " + JSON.stringify(error.debugInfo));

}

});

### getRange(rangeLocation: string)

Gets the whole body, or the starting or ending point of the body, as a range.

#### Syntax

bodyObject.getRange(rangeLocation);

#### Parameters

| **Parameter** | **Type** | **Description** |
| --- | --- | --- |
| rangeLocation | string | Optional. Optional. The range location can be 'Whole', 'Start' or 'End'. Possible values are: Whole, Start, End |

#### Returns

[Range](file:///C:\Users\chbigham\Documents\range.md)

### insertBreak(breakType: string, insertLocation: string)

Inserts a break at the specified location in the main document. The insertLocation value can be 'Start' or 'End'.

#### Syntax

bodyObject.insertBreak(breakType, insertLocation);

#### Parameters

| **Parameter** | **Type** | **Description** |
| --- | --- | --- |
| breakType | string | Required. The break type to add to the body. Possible values are: Page Page break at the insertion point.,Column Column break at the insertion point.,Next Section break on next page.,SectionContinuous New section without a corresponding page break.,SectionEven Section break with the next section beginning on the next even-numbered page. If the section break falls on an even-numbered page, Word leaves the next odd-numbered page blank.,SectionOdd Section break with the next section beginning on the next odd-numbered page. If the section break falls on an odd-numbered page, Word leaves the next even-numbered page blank.,Line Line break.,LineClearLeft Line break.,LineClearRight Line break.,TextWrapping Ends the current line and forces the text to continue below a picture, table, or other item. The text continues on the next blank line that does not contain a table aligned with the left or right margin. |
| insertLocation | string | Required. The value can be 'Start' or 'End'. Possible values are: Before Add content before the contents of the calling object.,After Add content after the contents of the calling object.,Start Prepend content to the contents of the calling object.,End Append content to the contents of the calling object.,Replace Replace the contents of the current object. |

#### Returns

void

#### Examples

// Run a batch operation against the Word object model.

Word.run(function (ctx) {

// Create a proxy object for the document body.

var body = ctx.document.body;

// Queue a commmand to insert a page break at the start of the document body.

body.insertBreak(Word.BreakType.page, Word.InsertLocation.start);

// Synchronize the document state by executing the queued commands,

// and return a promise to indicate task completion.

return ctx.sync().then(function () {

console.log('Added a page break at the start of the document body.');

});

})

.catch(function (error) {

console.log("Error: " + JSON.stringify(error));

if (error instanceof OfficeExtension.Error) {

console.log("Debug info: " + JSON.stringify(error.debugInfo));

}

});

### insertContentControl()

Wraps the body object with a Rich Text content control.

#### Syntax

bodyObject.insertContentControl();

#### Parameters

None

#### Returns

[ContentControl](file:///C:\Users\chbigham\Documents\contentcontrol.md)

#### Examples

// Run a batch operation against the Word object model.

Word.run(function (context) {

// Create a proxy object for the document body.

var body = context.document.body;

// Queue a commmand to wrap the body in a content control.

body.insertContentControl();

// Synchronize the document state by executing the queued commands,

// and return a promise to indicate task completion.

return context.sync().then(function () {

console.log('Wrapped the body in a content control.');

});

})

.catch(function (error) {

console.log('Error: ' + JSON.stringify(error));

if (error instanceof OfficeExtension.Error) {

console.log('Debug info: ' + JSON.stringify(error.debugInfo));

}

});

### insertFileFromBase64(base64File: string, insertLocation: string)

Inserts a document into the body at the specified location. The insertLocation value can be 'Replace', 'Start' or 'End'.

#### Syntax

bodyObject.insertFileFromBase64(base64File, insertLocation);

#### Parameters

| **Parameter** | **Type** | **Description** |
| --- | --- | --- |
| base64File | string | Required. The base64 encoded content of a .docx file. |
| insertLocation | string | Required. The value can be 'Replace', 'Start' or 'End'. Possible values are: Before Add content before the contents of the calling object.,After Add content after the contents of the calling object.,Start Prepend content to the contents of the calling object.,End Append content to the contents of the calling object.,Replace Replace the contents of the current object. |

#### Returns

[Range](file:///C:\Users\chbigham\Documents\range.md)

#### Examples

// Run a batch operation against the Word object model.

Word.run(function (context) {

// Create a proxy object for the document body.

var body = context.document.body;

// Queue a commmand to insert base64 encoded .docx at the beginning of the content body.

// You will need to implement getBase64() to pass in a string of a base64 encoded docx file.

body.insertFileFromBase64(getBase64(), Word.InsertLocation.start);

// Synchronize the document state by executing the queued commands,

// and return a promise to indicate task completion.

return context.sync().then(function () {

console.log('Added base64 encoded text to the beginning of the document body.');

});

})

.catch(function (error) {

console.log('Error: ' + JSON.stringify(error));

if (error instanceof OfficeExtension.Error) {

console.log('Debug info: ' + JSON.stringify(error.debugInfo));

}

});

### insertHtml(html: string, insertLocation: string)

Inserts HTML at the specified location. The insertLocation value can be 'Replace', 'Start' or 'End'.

#### Syntax

bodyObject.insertHtml(html, insertLocation);

#### Parameters

| **Parameter** | **Type** | **Description** |
| --- | --- | --- |
| html | string | Required. The HTML to be inserted in the document. |
| insertLocation | string | Required. The value can be 'Replace', 'Start' or 'End'. Possible values are: Before Add content before the contents of the calling object.,After Add content after the contents of the calling object.,Start Prepend content to the contents of the calling object.,End Append content to the contents of the calling object.,Replace Replace the contents of the current object. |

#### Returns

[Range](file:///C:\Users\chbigham\Documents\range.md)

#### Examples

// Run a batch operation against the Word object model.

Word.run(function (context) {

// Create a proxy object for the document body.

var body = context.document.body;

// Queue a commmand to insert HTML in to the beginning of the body.

body.insertHtml('<strong>This is text inserted with body.insertHtml()</strong>', Word.InsertLocation.start);

// Synchronize the document state by executing the queued commands,

// and return a promise to indicate task completion.

return context.sync().then(function () {

console.log('HTML added to the beginning of the document body.');

});

})

.catch(function (error) {

console.log('Error: ' + JSON.stringify(error));

if (error instanceof OfficeExtension.Error) {

console.log('Debug info: ' + JSON.stringify(error.debugInfo));

}

});

### insertInlinePictureFromBase64(base64EncodedImage: string, insertLocation: string)

Inserts a picture into the body at the specified location. The insertLocation value can be 'Start' or 'End'.

#### Syntax

bodyObject.insertInlinePictureFromBase64(base64EncodedImage, insertLocation);

#### Parameters

| **Parameter** | **Type** | **Description** |
| --- | --- | --- |
| base64EncodedImage | string | Required. The base64 encoded image to be inserted in the body. |
| insertLocation | string | Required. The value can be 'Start' or 'End'. Possible values are: Before Add content before the contents of the calling object.,After Add content after the contents of the calling object.,Start Prepend content to the contents of the calling object.,End Append content to the contents of the calling object.,Replace Replace the contents of the current object. |

#### Returns

[InlinePicture](file:///C:\Users\chbigham\Documents\inlinepicture.md)

#### Examples

// Run a batch operation against the Word object model.

Word.run(function (context) {

// Create a proxy object for the document body.

var body = context.document.body;

// Queue a commmand to insert OOXML in to the beginning of the body.

body.insertOoxml("<pkg:package xmlns:pkg='http://schemas.microsoft.com/office/2006/xmlPackage'><pkg:part pkg:name='/\_rels/.rels' pkg:contentType='application/vnd.openxmlformats-package.relationships+xml' pkg:padding='512'><pkg:xmlData><Relationships xmlns='http://schemas.openxmlformats.org/package/2006/relationships'><Relationship Id='rId1' Type='http://schemas.openxmlformats.org/officeDocument/2006/relationships/officeDocument' Target='word/document.xml'/></Relationships></pkg:xmlData></pkg:part><pkg:part pkg:name='/word/document.xml' pkg:contentType='application/vnd.openxmlformats-officedocument.wordprocessingml.document.main+xml'><pkg:xmlData><w:document xmlns:w='http://schemas.openxmlformats.org/wordprocessingml/2006/main' ><w:body><w:p><w:pPr><w:spacing w:before='360' w:after='0' w:line='480' w:lineRule='auto'/><w:rPr><w:color w:val='70AD47' w:themeColor='accent6'/><w:sz w:val='28'/></w:rPr></w:pPr><w:r><w:rPr><w:color w:val='70AD47' w:themeColor='accent6'/><w:sz w:val='28'/></w:rPr><w:t>This text has formatting directly applied to achieve its font size, color, line spacing, and paragraph spacing.</w:t></w:r></w:p></w:body></w:document></pkg:xmlData></pkg:part></pkg:package>", Word.InsertLocation.start);

// Synchronize the document state by executing the queued commands,

// and return a promise to indicate task completion.

return context.sync().then(function () {

console.log('OOXML added to the beginning of the document body.');

});

})

.catch(function (error) {

console.log('Error: ' + JSON.stringify(error));

if (error instanceof OfficeExtension.Error) {

console.log('Debug info: ' + JSON.stringify(error.debugInfo));

}

});

Additional information

Read [Create better add-ins for Word with Office Open XML](https://msdn.microsoft.com/en-us/library/office/dn423225.aspx) for guidance on working with OOXML. The [Word-Add-in-DocumentAssembly][body.insertOoxml] sample shows how you can use this API to assemble a document.

### insertOoxml(ooxml: string, insertLocation: string)

Inserts OOXML at the specified location. The insertLocation value can be 'Replace', 'Start' or 'End'.

#### Syntax

bodyObject.insertOoxml(ooxml, insertLocation);

#### Parameters

| **Parameter** | **Type** | **Description** |
| --- | --- | --- |
| ooxml | string | Required. The OOXML to be inserted. |
| insertLocation | string | Required. The value can be 'Replace', 'Start' or 'End'. Possible values are: Before Add content before the contents of the calling object.,After Add content after the contents of the calling object.,Start Prepend content to the contents of the calling object.,End Append content to the contents of the calling object.,Replace Replace the contents of the current object. |

#### Returns

[Range](file:///C:\Users\chbigham\Documents\range.md)

### insertParagraph(paragraphText: string, insertLocation: string)

Inserts a paragraph at the specified location. The insertLocation value can be 'Start' or 'End'.

#### Syntax

bodyObject.insertParagraph(paragraphText, insertLocation);

#### Parameters

| **Parameter** | **Type** | **Description** |
| --- | --- | --- |
| paragraphText | string | Required. The paragraph text to be inserted. |
| insertLocation | string | Required. The value can be 'Start' or 'End'. Possible values are: Before Add content before the contents of the calling object.,After Add content after the contents of the calling object.,Start Prepend content to the contents of the calling object.,End Append content to the contents of the calling object.,Replace Replace the contents of the current object. |

#### Returns

[Paragraph](file:///C:\Users\chbigham\Documents\paragraph.md)

#### Examples

// Run a batch operation against the Word object model.

Word.run(function (context) {

// Create a proxy object for the document body.

var body = context.document.body;

// Queue a commmand to insert the paragraph at the end of the document body.

body.insertParagraph('Content of a new paragraph', Word.InsertLocation.end);

// Synchronize the document state by executing the queued commands,

// and return a promise to indicate task completion.

return context.sync().then(function () {

console.log('Paragraph added at the end of the document body.');

});

})

.catch(function (error) {

console.log('Error: ' + JSON.stringify(error));

if (error instanceof OfficeExtension.Error) {

console.log('Debug info: ' + JSON.stringify(error.debugInfo));

}

});

Additional information The [Word-Add-in-DocumentAssembly][body.insertParagraph] sample shows how you can use the insertParagraph method to assemble a document.

### insertTable(rowCount: number, columnCount: number, insertLocation: string, values: string[][])

Inserts a table with the specified number of rows and columns. The insertLocation value can be 'Start' or 'End'.

#### Syntax

bodyObject.insertTable(rowCount, columnCount, insertLocation, values);

#### Parameters

| **Parameter** | **Type** | **Description** |
| --- | --- | --- |
| rowCount | number | Required. The number of rows in the table. |
| columnCount | number | Required. The number of columns in the table. |
| insertLocation | string | Required. The value can be 'Start' or 'End'. Possible values are: Before Add content before the contents of the calling object.,After Add content after the contents of the calling object.,Start Prepend content to the contents of the calling object.,End Append content to the contents of the calling object.,Replace Replace the contents of the current object. |
| values | string[][] | Optional. Optional 2D array. Cells are filled if the corresponding strings are specified in the array. |

#### Returns

[Table](file:///C:\Users\chbigham\Documents\table.md)

### insertText(text: string, insertLocation: string)

Inserts text into the body at the specified location. The insertLocation value can be 'Replace', 'Start' or 'End'.

#### Syntax

bodyObject.insertText(text, insertLocation);

#### Parameters

| **Parameter** | **Type** | **Description** |
| --- | --- | --- |
| text | string | Required. Text to be inserted. |
| insertLocation | string | Required. The value can be 'Replace', 'Start' or 'End'. Possible values are: Before Add content before the contents of the calling object.,After Add content after the contents of the calling object.,Start Prepend content to the contents of the calling object.,End Append content to the contents of the calling object.,Replace Replace the contents of the current object. |

#### Returns

[Range](file:///C:\Users\chbigham\Documents\range.md)

#### Examples

// Run a batch operation against the Word object model.

Word.run(function (context) {

// Create a proxy object for the document body.

var body = context.document.body;

// Queue a commmand to insert text in to the beginning of the body.

body.insertText('This is text inserted with body.insertText()', Word.InsertLocation.start);

// Synchronize the document state by executing the queued commands,

// and return a promise to indicate task completion.

return context.sync().then(function () {

console.log('Text added to the beginning of the document body.');

});

})

.catch(function (error) {

console.log('Error: ' + JSON.stringify(error));

if (error instanceof OfficeExtension.Error) {

console.log('Debug info: ' + JSON.stringify(error.debugInfo));

}

});

### load(param: object)

Fills the proxy object created in JavaScript layer with property and object values specified in the parameter.

#### Syntax

object.load(param);

#### Parameters

| **Parameter** | **Type** | **Description** |
| --- | --- | --- |
| param | object | Optional. Accepts parameter and relationship names as delimited string or an array. Or, provide [loadOption](file:///C:\Users\chbigham\Documents\loadoption.md) object. |

#### Returns

void

#### Examples

// Run a batch operation against the Word object model.

Word.run(function (context) {

// Create a proxy object for the document body.

var body = context.document.body;

// Queue a commmand to load font and style information for the document body.

context.load(body, 'font/size, font/name, font/color, style');

// Synchronize the document state by executing the queued commands,

// and return a promise to indicate task completion.

return context.sync().then(function () {

// Show the results of the load method. Here we show the

// property values on the body object.

var results = 'Font size: ' + body.font.size +

'; Font name: ' + body.font.name +

'; Font color: ' + body.font.color +

'; Body style: ' + body.style;

console.log(results);

});

})

.catch(function (error) {

console.log('Error: ' + JSON.stringify(error));

if (error instanceof OfficeExtension.Error) {

console.log('Debug info: ' + JSON.stringify(error.debugInfo));

}

});

### search(searchText: string, searchOptions: ParamTypeStrings.SearchOptions)

Performs a search with the specified searchOptions on the scope of the body object. The search results are a collection of range objects.

#### Syntax

bodyObject.search(searchText, searchOptions);

#### Parameters

| **Parameter** | **Type** | **Description** |
| --- | --- | --- |
| searchText | string | Required. The search text. |
| searchOptions | ParamTypeStrings.SearchOptions | Optional. Optional. Options for the search. |

#### Returns

[SearchResultCollection](file:///C:\Users\chbigham\Documents\searchresultcollection.md)

#### Examples

// Run a batch operation against the Word object model.

Word.run(function (context) {

// Create a proxy object for the document body.

var body = context.document.body;

// Queue a commmand to search the document.

var searchResults = context.document.body.search('video', {matchCase: false});

// Queue a commmand to load the results.

context.load(searchResults, 'text, font');

// Synchronize the document state by executing the queued commands,

// and return a promise to indicate task completion.

return context.sync().then(function () {

var results = 'Found count: ' + searchResults.items.length +

'; we highlighted the results.';

// Queue a command to change the font for each found item.

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

searchResults.items[i].font.color = '#FF0000' // Change color to Red

searchResults.items[i].font.highlightColor = '#FFFF00';

searchResults.items[i].font.bold = true;

}

// Synchronize the document state by executing the queued commands,

// and return a promise to indicate task completion.

return context.sync().then(function () {

console.log(results);

});

});

})

.catch(function (error) {

console.log('Error: ' + JSON.stringify(error));

if (error instanceof OfficeExtension.Error) {

console.log('Debug info: ' + JSON.stringify(error.debugInfo));

}

});

Additional information The [Word-Add-in-DocumentAssembly][body.search] sample provides another example of how to search a document.

### select(selectionMode: string)

Selects the body and navigates the Word UI to it.

#### Syntax

bodyObject.select(selectionMode);

#### Parameters

| **Parameter** | **Type** | **Description** |
| --- | --- | --- |
| selectionMode | string | Optional. Optional. The selection mode can be 'Select', 'Start' or 'End'. 'Select' is the default. Possible values are: Select, Start, End |

#### Returns

void

#### Examples

// Run a batch operation against the Word object model.

Word.run(function (context) {

// Create a proxy object for the document body.

var body = context.document.body;

// Queue a commmand to select the document body. The Word UI will

// move to the selected document body.

body.select();

// Synchronize the document state by executing the queued commands,

// and return a promise to indicate task completion.

return context.sync().then(function () {

console.log('Selected the document body.');

});

})

.catch(function (error) {

console.log('Error: ' + JSON.stringify(error));

if (error instanceof OfficeExtension.Error) {

console.log('Debug info: ' + JSON.stringify(error.debugInfo));

}

});

### Property access examples

Get the text property on the body object

// Run a batch operation against the Word object model.

Word.run(function (context) {

// Create a proxy object for the document body.

var body = context.document.body;

// Queue a commmand to load the text in document body.

context.load(body, 'text');

// Synchronize the document state by executing the queued commands,

// and return a promise to indicate task completion.

return context.sync().then(function () {

console.log("Body contents: " + body.text);

});

})

.catch(function (error) {

console.log("Error: " + JSON.stringify(error));

if (error instanceof OfficeExtension.Error) {

console.log("Debug info: " + JSON.stringify(error.debugInfo));

}

});

Get the style and the font size, font name, and font color properties on the body object.

// Run a batch operation against the Word object model.

Word.run(function (context) {

// Create a proxy object for the document body.

var body = context.document.body;

// Queue a commmand to load font and style information for the document body.

context.load(body, 'font/size, font/name, font/color, style');

// Synchronize the document state by executing the queued commands,

// and return a promise to indicate task completion.

return context.sync().then(function () {

// Show the results of the load method. Here we show the

// property values on the body object.

var results = 'Font size: ' + body.font.size +

'; Font name: ' + body.font.name +

'; Font color: ' + body.font.color +

'; Body style: ' + body.style;

console.log(results);

});

})

.catch(function (error) {

console.log('Error: ' + JSON.stringify(error));

if (error instanceof OfficeExtension.Error) {

console.log('Debug info: ' + JSON.stringify(error.debugInfo));

}

});