# Text

- Edit

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Chr</code>(Integer)</td>
<td>Returns a single-character string corresponding to the 'c' character code.</td>
</tr>
<tr>
<td><code>Concat</code>(Text, Text)</td>
<td>Returns the concatenation of two Texts: 't1' and 't2'.</td>
</tr>
<tr>
<td><code>EncodeHtml</code>(Text)</td>
<td>Replaces special characters in a string so that you can use it in HTML literals. Use this function when using un-escaped expressions that contain content provided by end users. Warning: Since this function only encodes strings that will be used in HTML literals, it does not protect you from cross-site scripting (XSS) or JavaScript injection vulnerabilities on its own. <strong>Do not</strong> use this function to encode text that might get executed as JavaScript code, only to encode HTML literals.</td>
</tr>
<tr>
<td><code>EncodeJavaScript</code>(Text)</td>
<td>Replaces special characters in a string so that you can use it in JavaScript literals. Use this function when using un-escaped expressions that contain content provided by end users. Warning: Since this function only encodes strings that will be used in JavaScript literals, it does not protect you from cross-site scripting (XSS) or JavaScript injection vulnerabilities on its own. <strong>Do not</strong> use this function to encode text that might get executed as JavaScript code, only to encode JavaScript literals.</td>
</tr>
<tr>
<td><code>EncodeSql</code>(Text)</td>
<td>Replaces special characters in a string literal so that you can use it in a SQL statement. Use this function when the Expand Inline property of a Query Parameter is enabled to escape content provided by end users. Warning: Since this function only encodes string literals, it does not protect you from SQL injection vulnerabilities on its own. <strong>Do not</strong> use this function to encode text that might get executed as part of the SQL statement. Check the OutSystems Best Practices documentation for more information on building dynamic SQL statements the right way.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>EncodeUrl</strong> <em>(Text)</em></td>
<td>Replaces all non-alphanumeric characters in a string, i.e. characters outside of the [0-9a-zA-Z] range, so that you can safely use it in URL parameter values. Use this function to build URLs in your application that may contain content provided by end users, e.g. when dynamically building URLs to an external site.</td>
</tr>
<tr>
<td><strong>Index</strong> <em>(Text, Text, Integer, Boolean, Boolean)</em></td>
<td>Returns the zero-based position in Text ‘t’ where ‘search’ Text can be found. Returns -1 if ‘search’ is not found or if ‘search’ is empty.</td>
</tr>
<tr>
<td><strong>Length</strong> <em>(Text)</em></td>
<td>Returns the number of characters in Text ‘t’.</td>
</tr>
<tr>
<td><strong>NewLine</strong> <em>(Text)</em></td>
<td>Returns a string containing the New Line (Return) character.</td>
</tr>
<tr>
<td><strong>Replace</strong> <em>(Text, Text, Text)</em></td>
<td>Returns Text ‘t’ after replacing all Text occurrences of ‘search’ with ‘replace’.</td>
</tr>
<tr>
<td><strong>Substr</strong> <em>(Text, Integer, Integer)</em></td>
<td>Returns a sub-string of ‘t’ beginning at ‘start’ zero-based position and with ‘length’ characters.</td>
</tr>
<tr>
<td><strong>ToLower</strong> <em>(Text)</em></td>
<td>Converts Text ‘t’ to the equivalent lowercase text.</td>
</tr>
<tr>
<td><strong>ToUpper</strong> <em>(Text)</em></td>
<td>Converts Text ‘t’ to the equivalent uppercase text.</td>
</tr>
<tr>
<td><strong>Trim</strong> <em>(Text)</em></td>
<td>Removes all leading and trailing space characters (‘ ’) from Text ‘t’.</td>
</tr>
<tr>
<td><strong>TrimEnd</strong> <em>(Text)</em></td>
<td>Removes all trailing space characters (‘ ’) from Text ‘t’.</td>
</tr>
<tr>
<td><strong>TrimStart</strong> <em>(Text)</em></td>
<td>Removes all leading space characters (‘ ’) from Text ‘t’.</td>
</tr>
</tbody>
</table>

**Chr**

Returns a single-character string corresponding to the ‘c’ character code.

Available in:

- Server-side logic: Yes
- Client-side logic: Yes
- Database: Function is evaluated before the aggregate is executed.
- Local Storage: Function is evaluated before the aggregate is executed.
Parameters

c
Type: Integer. Mandatory.
The ASCII code value to be converted to a character.

Output

Type: Text

Examples

| Chr(88) = "X" |

Concat

Returns the concatenation of two Texts: 't1' and 't2'.

Available in:

- Server-side logic: Yes
- Client-side logic: Yes
- Database: Can be used with attributes in aggregates.
- Local Storage: Can be used with attributes in aggregates.

Parameters

\( t1 \)
Type: Text. Mandatory.
The first string.

**t2**
Type: Text. Mandatory.
The string that will be appended to the first string in the output.

### Output
Type: Text

### Examples

```
Concat("First string", "last string") = "First stringlast string"
Concat("", "") = ""
```

### EncodeHtml
Replaces special characters in a string so that you can use it in HTML literals. Use this function when using un-escaped expressions that contain content provided by end users.

Warning: Since this function only encodes strings that will be used in HTML literals, it does not protect you from cross-site scripting (XSS) or JavaScript injection vulnerabilities on its own. **Do not** use this function to encode text that might get executed as JavaScript code, only to encode HTML literals.

Available in:

- Server-side logic: Yes
- Client-side logic: Yes
- Database: Function is evaluated before the aggregate is executed.
- Local Storage: Function is evaluated before the aggregate is executed.
Parameters

text
Type: Text. Mandatory.
The Text to be encoded.

Output

Type: Text

Examples

EncodeHtml("<>") = "&lt;&gt;"
EncodeHtml("another ' test") = "another &apos; test"
EncodeHtml("another " test") = "another " test"
EncodeHtml("Hello" + NewLine() + "World!") = "Hello<br/>World!"

// Usage in an Expression with Escape Content = No:
Value = "<dl><dt>" + EncodeHtml(ArticleTitle) + "</dt><dd>" +
        EncodeHtml(ArticleDescription) + "</dd></dl>"

EncodeJavaScript

Replaces special characters in a string so that you can use it in JavaScript literals. Use this function when using un-escaped expressions that contain content provided by end users.

Warning: Since this function only encodes strings that will be used in JavaScript literals, it does not protect you from cross-site scripting (XSS) or JavaScript injection vulnerabilities on its own. Do not use this function to encode text that might get executed as JavaScript code, only to encode JavaScript literals.

Available in:

- Server-side logic: Yes
- Client-side logic: Yes
- Database: Function is evaluated before the aggregate is executed.
- Local Storage: Function is evaluated before the aggregate is executed.

### Parameters

**text**

Type: Text. Mandatory.
The text to be encoded.

### Output

Type: Text

### Examples

```plaintext
EncodeJavaScript("another ' test") = "another \x27 test"
EncodeJavaScript("<>") = "\x3c\x3e"

// Usage: Defining the Script property of the RunJavaScript action from HTTPRequest Handler.
// 'Title' contains user entered content.
Script = "ChangeContainerContent('" + ModalTitle.Id + ", ", " + EncodeJavaScript(Title) + ");"
```

### EncodeSql

Replaces special characters in a string literal so that you can use it in a SQL statement. Use this function when the Expand Inline property of a Query Parameter is enabled to escape content provided by end users.
Warning: Since this function only encodes string literals, it does not protect you from SQL injection vulnerabilities on its own. **Do not** use this function to encode text that might get executed as part of the SQL statement. Check the OutSystems Best Practices documentation for more information on building dynamic SQL statements the right way.

Available in:

- Server-side logic: Yes
- Client-side logic: No
- Database: Function is evaluated before the aggregate is executed.
- Local Storage: Function is evaluated before the aggregate is executed.

### Parameters

**text**

Type: Text. Mandatory.

The Text to be encoded.

### Output

Type: Text

### Examples

```javascript
EncodeSql("another ' test") = "another '' test"
```

// Usage in SQL element:
Statement = SELECT {Users}.{Username}, {Users}.{Firstname}, {Users}.{Lastname} FROM {Users} WHERE {Users}.{IsActive} = 1 @ extraFilters

// ... where the extraFilters query parameter would encode the literal provided by the user:
extraFilters = If(lastnameFilter <> "", "AND [Users].{Lastname} like '%$' + EncodeSql(lastnameFilter) + '%$', "")
EncodeUrl

Replaces all non-alphanumeric characters in a string, i.e. characters outside of the [0-9a-zA-Z] range, so that you can safely use it in URL parameter values. Use this function to build URLs in your application that may contain content provided by end users, e.g. when dynamically building URLs to an external site.

Available in:

- Server-side logic: Yes
- Client-side logic: Yes
- Database: Function is evaluated before the aggregate is executed.
- Local Storage: Function is evaluated before the aggregate is executed.

Parameters

text
Type: Text. Mandatory.
The Text to be encoded.

Output

Type: Text

Examples

```plaintext
EncodeUrl(" test") = "+test"
EncodeUrl("another ' test") = "another+%27+test"
EncodeUrl("<>") = "%3c%3e"
EncodeUrl("1+2") = "1%2b2"
```
EncodeUrl("Company A&A") = "Company+A%26A"

// Usage when building an external URL:
ExternalUrl = "http://www.example.com/?company=" + EncodeUrl(CompanyName)

Index

Returns the zero-based position in Text 't' where 'search' Text can be found. Returns -1 if 'search' is not found or if 'search' is empty.

Available in:

- Server-side logic: Yes
- Client-side logic: Yes
- Database: Can be used with attributes in aggregates.
- Local Storage: Can be used with attributes in aggregates.

Parameters

t
  Type: Text. Mandatory.
  The Text where the search Text can be found.

search
  Type: Text. Mandatory.
  The Text string to be found.

startIndex
  Type: Integer.
  Indicates the (zero-based) index where the search starts. In case of searching from the end to the start, a startIndex different from 0 (zero) indicates the end of the text. The default value is 0 (zero). When used in Aggregates this parameter is not present.

searchFromEnd
  Type: Boolean.
  Indicates the direction of the search. In case of searching from the end to the start, a startIndex different from 0 (zero) indicates the end of the text. The default value is False. When used in Aggregates this parameter is not present.
**ignoreCase**
Type: Boolean.

**Output**
Type: Integer

**Examples**

Index("First string", "F") = 0
Index("First string", "st") = 3
Index("First string", "xx") = -1
Index("First string", "F", startIndex: 5) = -1
Index("First string", "st", startIndex: 5) = 6
Index("First string", "xx", startIndex: 5) = -1
Index("First string", "F", searchFromEnd: True) = 0
Index("First string", "st", searchFromEnd: True) = 6
Index("First string", "xx", searchFromEnd: True) = -1
Index("First string", "f") = -1
Index("First string", "f", ignoreCase: True) = 0
Index("", "xx") = -1
Index("First string", "") = -1
Index("", ") = -1

**Length**

Returns the number of characters in Text 't'.

Available in:

- Server-side logic: Yes
- Client-side logic: Yes
- Database: Can be used with attributes in aggregates.
- Local Storage: Can be used with attributes in aggregates.

Updated: Sat, 04 Jul 2020 13:30:25 GMT
Parameters

\( t \)

Type: Text. Mandatory.
The Text to calculate the length of.

Output

Type: Integer

Examples

\[
\begin{align*}
\text{Length("First string")} &= 12 \\
\text{Length("")} &= 0
\end{align*}
\]

NewLine

Returns a string containing the New Line (Return) character.

Available in:

- Server-side logic: Yes
- Client-side logic: Yes
- Database: Function is evaluated before the aggregate is executed.
- Local Storage: Function is evaluated before the aggregate is executed.
Replace

Returns Text 't' after replacing all Text occurrences of 'search' with 'replace'.

Available in:

- Server-side logic: Yes
- Client-side logic: Yes
- Database: Can be used with attributes in aggregates.
- Local Storage: Can be used with attributes in aggregates.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type: Text. Mandatory.</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td>The Text where the search and replace operation is performed.</td>
</tr>
<tr>
<td>search</td>
<td>The Text to be replaced.</td>
</tr>
<tr>
<td>replace</td>
<td>The Text that replaces the search Text.</td>
</tr>
</tbody>
</table>

Output

Type: Text
Examples

Replace("First string", "xx", ") = "First string"
Replace("First string", "First", "Second") = "Second string"
Replace("First string", "First", ") = " string"

Substr

Returns a sub-string of 't' beginning at 'start' zero-based position and with 'length' characters.

Available in:

- Server-side logic: Yes
- Client-side logic: Yes
- Database: Can be used with attributes in aggregates.
- Local Storage: Can be used with attributes in aggregates.

Parameters

- **t**
  Type: Text. Mandatory.
  The Text where the operation is performed.

- **start**
  Type: Integer. Mandatory.
  The zero-based position to start the Text extraction from.

- **length**
  Type: Integer. Mandatory.
  The number of characters to include in the output Text.
Substr("First string", 2, 4) = "rst \\
Substr("First string", 0, 100) = "First string" \\
Substr("First string", 11, 3) = "g" \\
Substr("First string", Length("First string"), 0) = "" \\
Substr("First string", 2, 0) = ""

ToLower

Converts Text 't' to the equivalent lowercase text.

Available in:

- Server-side logic: Yes
- Client-side logic: Yes
- Database: Can be used with attributes in aggregates.
- Local Storage: Can be used with attributes in aggregates.

Parameters

t

Type: Text. Mandatory.
Thet Text to transform into lowercase.
Examples

ToLower("First string") = "first string"

ToUpper

Converts Text 't' to the equivalent uppercase text.

Available in:

- Server-side logic: Yes
- Client-side logic: Yes
- Database: Can be used with attributes in aggregates.
- Local Storage: Can be used with attributes in aggregates.

Parameters

\( t \)

Type: Text. Mandatory.
The Text to transform into uppercase.

Output

Type: Text
Examples

ToUpper("First string") = "FIRST STRING"

Trim

Removes all leading and trailing space characters (' ') from Text 't'.

Available in:

- Server-side logic: Yes
- Client-side logic: Yes
- Database: Can be used with attributes in aggregates.
- Local Storage: Can be used with attributes in aggregates.

Parameters

<table>
<thead>
<tr>
<th>t</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type: Text. Mandatory.</td>
</tr>
<tr>
<td>The Text to trim.</td>
</tr>
</tbody>
</table>

Output

Type: Text

Examples

Trim(" First string ") = "First string"
Trim("First string ") = "First string"
TrimEnd

Removes all trailing space characters (" ") from Text 't'.

Available in:

• Server-side logic: Yes
• Client-side logic: Yes
• Database: Can be used with attributes in aggregates.
• Local Storage: Can be used with attributes in aggregates.

Parameters

\[ t \]
Type: Text. Mandatory.
The Text to remove trailing space characters from.

Output

Type: Text

Examples

\[
\begin{align*}
\text{TrimEnd(" First string ") } &= " \text{ First string} \\
\text{TrimEnd("First string ") } &= "\text{First string"
\end{align*}
\]
TrimStart

Removes all leading space characters (" ") from Text 't'.

Available in:

- Server-side logic: Yes
- Client-side logic: Yes
- Database: Can be used with attributes in aggregates.
- Local Storage: Can be used with attributes in aggregates.

Parameters

t

Type: Text. Mandatory.
The Text to remove leading space characters from.

Output

Type: Text

Examples

TrimStart(" First string ") = "First string 
TrimStart("First string ") = "First string 