Learning to Escape Double Quotes in Google Sheets Formulas

Authored by

Mohammed looti

March 7, 2026

When processing or generating structured data within Google Sheets, developers and advanced users frequently encounter the critical necessity of embedding literal double quotes within a text string, typically as part of a complex formula. This requirement stems from the fundamental rule that double quotes serve a dual purpose: they denote the beginning and end of a text string itself. If you attempt to use a quote character directly inside a string, Google Sheets will prematurely terminate the string, resulting in a syntax error or an incorrect calculation.

The technique used to overcome this parsing conflict is known as escaping quotes. Proper escaping is essential for maintaining data integrity, especially when preparing data for systems that strictly require quoted values, such as CSV (Comma Separated Values) exports, SQL queries, or JSON structures. Without correctly implemented escaping, your output data may be corrupted or unreadable by downstream applications, rendering your entire spreadsheet process unreliable.

Fortunately, Google Sheets provides two distinct, powerful, and highly effective methods for resolving this challenge. Both approaches enable you to embed the necessary double quote character within a larger string, ensuring the text is interpreted correctly as a literal character rather than a syntax delimiter. Understanding these techniques is paramount for anyone involved in advanced data manipulation or integration tasks within the spreadsheet environment. We will thoroughly explore both methods, providing clear rationale and practical, step-by-step examples.

Method 1: Leveraging the "Doubled Quotes" Escape Sequence

The first method, often favored for its conciseness, involves using a sequence of double quotes to represent a single literal quote. In the context of Google Sheets formulas and many other programming languages, placing four consecutive double quotes (`""""`) within a formula string literal is interpreted by the parser as a request for one single double quote character. The outer two quotes define the literal string, while the inner pair acts as the escape sequence for the quote character itself.

This technique is most effectively employed alongside the CONCATENATE function, which is designed to join two or more text strings into one single output string. To successfully wrap the content of a cell, such as **A2**, in quotes, you must concatenate the opening quote escape sequence (`""""`), the cell reference (A2), and the closing quote escape sequence (`""""`). This combination ensures that the resulting output is a cohesive, quoted unit.

The structure of the formula using this intuitive, if visually dense, method is presented below. This approach is powerful because it relies solely on the core syntax of formulas rather than specialized functions, offering high portability across different spreadsheet environments.

=CONCATENATE(""""",A2,""""")

Method 2: Utilizing the CHAR(34) Function for Clarity

For users who prioritize explicit code readability over conciseness, the CHAR function offers a highly transparent alternative for inserting the double quote character. The CHAR function is designed to convert a numeric code into the corresponding character based on the current system's character set, typically Unicode.

Specifically, the Unicode numeric value for the double quote character is **34**. Therefore, the expression CHAR(34) explicitly and unambiguously represents the required quote character. This method eliminates the potential confusion arising from the repetitive quote marks in the first technique, making the formula easier to audit and maintain, particularly for complex spreadsheet projects.

Just like Method 1, this approach requires the use of the CONCATENATE function (or the ampersand operator, although CONCATENATE is often preferred for structure) to combine the opening quote, the target string (e.g., A2), and the closing quote. The resulting formula clearly outlines the intent: append character 34, insert the cell content, and then append character 34 again.

The formula structure for utilizing the CHAR(34) function is detailed here:

=CONCATENATE(CHAR(34),A2,CHAR(34))

Practical Application: Preparing Data for External Systems

To fully appreciate the importance of quote escaping, let us apply both methods to a typical, real-world scenario. Imagine you have a dataset containing product names, descriptive text, or identifiers in Google Sheets. These raw strings must be exported into a database or a specific software application that mandates that all text fields be enclosed within double quotes to prevent parsing issues, especially if the text contains commas or other delimiters.

Consider the following list of items residing in Column A of your spreadsheet, starting from the cell **A2**. Our goal is not just to display the content but to generate new, quoted strings in Column B that are ready for external consumption.

	A	B	C	D
1	**String**			
2	Hello there everyone			
3	Hey what is going on			
4	How is everybody doing			
5	Hi how are you			
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				

We will systematically apply both the "Doubled Quotes" technique and the CHAR(34) function to this sample data. By executing both procedures, we will demonstrate that while the underlying logic and syntax differ, the final, correctly formatted output string remains identical, affirming the reliability of both methods.

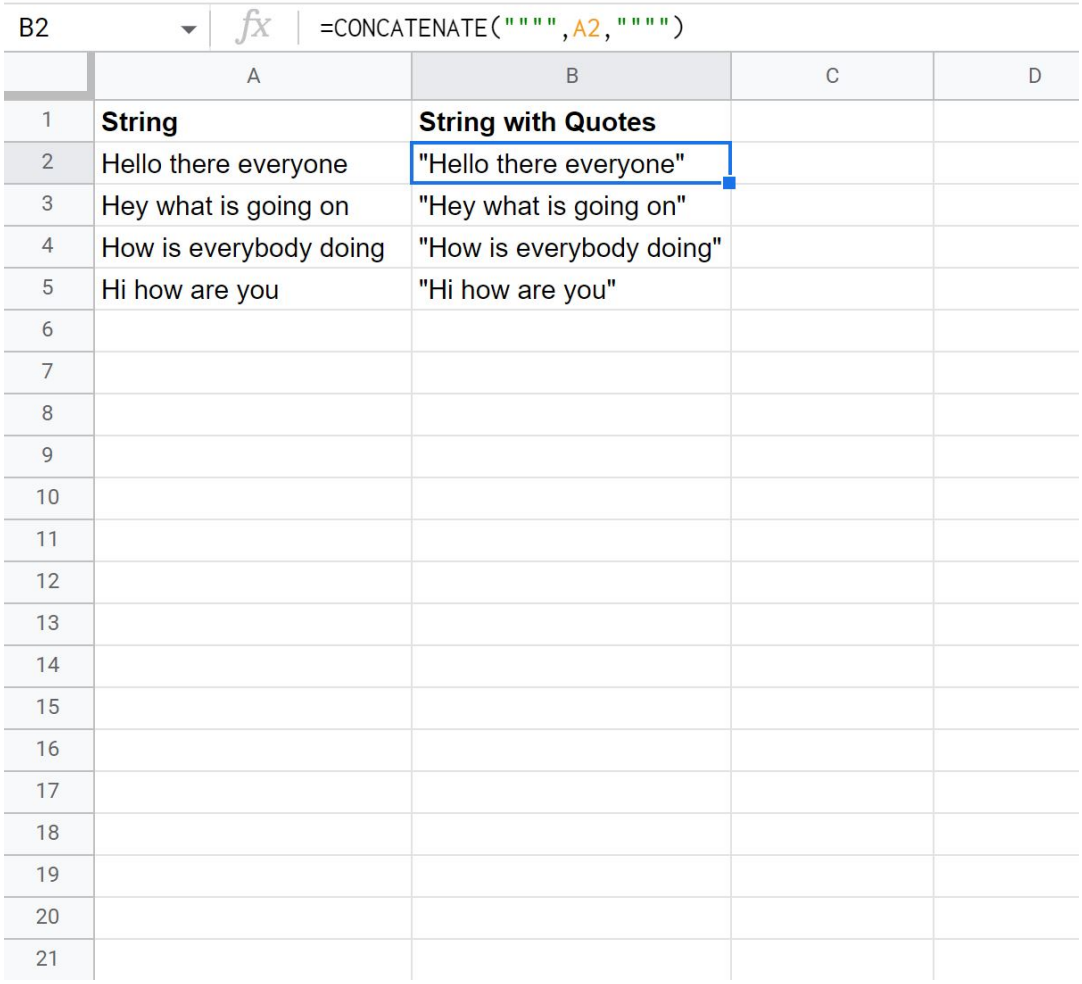
Step-by-Step Example 1: Implementing the Doubled Quotes Method

We initiate the process by applying Method 1 to the first data point in cell **A2**. The objective is to produce the required quoted string in the adjacent cell, **B2**. This single formula will then be propagated across the rest of the column.

To achieve this, type the following formula directly into cell **B2**:

=CONCATENATE("""",A2,"""")

Upon pressing Enter, the cell **B2** will immediately display the content of **A2** correctly wrapped in double quotes. To apply this transformation efficiently to the entire dataset, simply use the drag and fill handle located at the bottom right corner of cell B2, dragging it down the column corresponding to your data range.



	A	B	C	D
1	**String**	**String with Quotes**		
2	Hello there everyone	"Hello there everyone"		
3	Hey what is going on	"Hey what is going on"		
4	How is everybody doing	"How is everybody doing"		
5	Hi how are you	"Hi how are you"		
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				
21				

As clearly illustrated in the image above, every resulting cell in Column B now holds a string that includes the necessary literal double quotes. This confirms that the "Doubled Quotes" method is a successful and streamlined solution for generating quoted data structures from raw spreadsheet inputs.

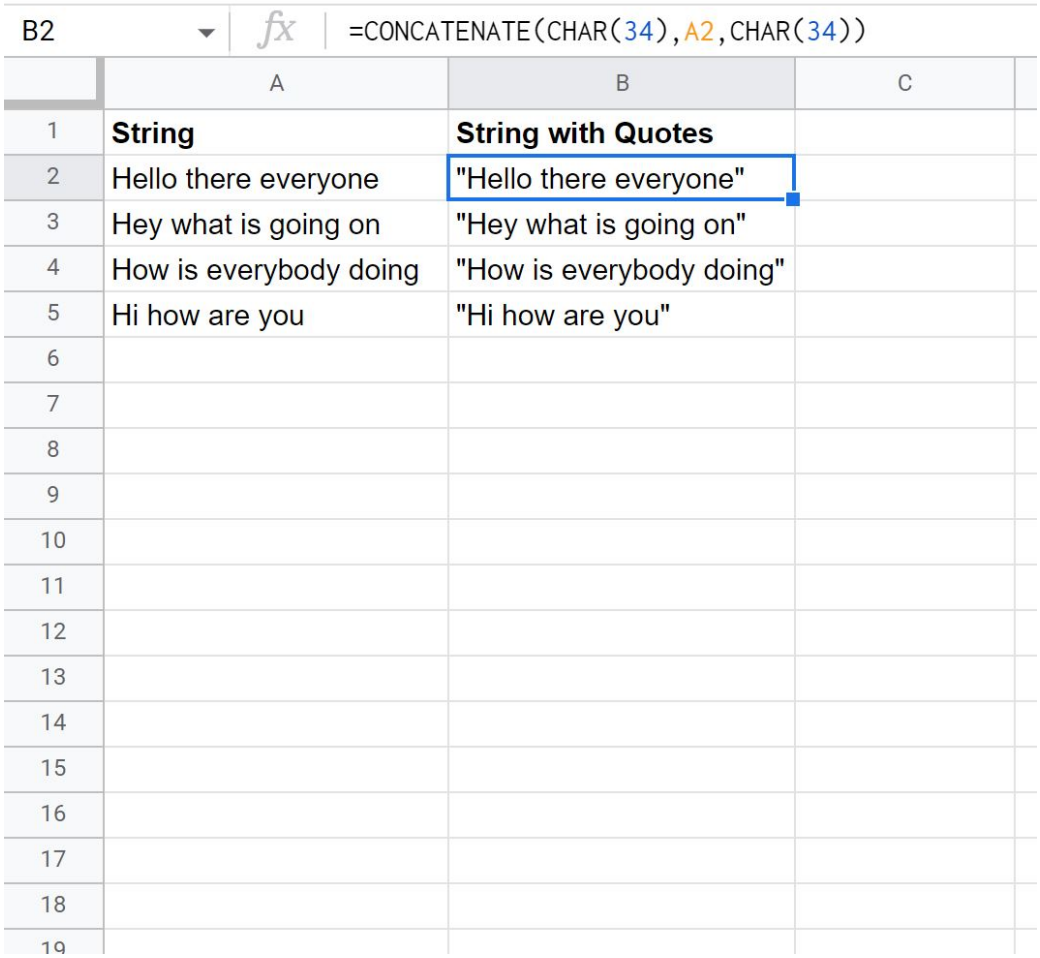
Step-by-Step Example 2: Implementing the CHAR(34) Method

We will now perform the identical operation using Method 2, employing the CHAR(34) function. This demonstration serves to highlight the flexibility within Google Sheets and offers an important comparison between the two syntactical styles. We will place the result in the same output column, beginning again in cell **B2**.

Enter the following highly readable formula into cell **B2**:

=CONCATENATE(CHAR(34),A2,CHAR(34))

After confirming the formula entry, use the drag and fill functionality to apply this change down the entirety of Column B. This action instantaneously processes the raw data from Column A, ensuring every entry is formatted according to the strict requirements of external systems.



	A	B	C
1	**String**	**String with Quotes**	
2	Hello there everyone	"Hello there everyone"	
3	Hey what is going on	"Hey what is going on"	
4	How is everybody doing	"How is everybody doing"	
5	Hi how are you	"Hi how are you"	
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			

The resulting dataset, pictured above, confirms the identical output compared to the first method. The key advantage of using CHAR(34) is its explicit reference to the Unicode character code, which enhances formula transparency and debuggability for those who prefer to work with character encoding standards. This reinforces the principle that while both methods are functionally equivalent, they cater to different preferences for formula construction and legibility.

Conclusion: Choosing Your Quote Escaping Strategy

Successfully mastering the technique of escaping quotes in Google Sheets is a fundamental requirement for anyone performing sophisticated data preparation or integration tasks. We have thoroughly examined the two most reliable methods available: the concise "Doubled Quotes" technique using the `""""""` sequence and the explicit, character-code-based approach utilizing the CHAR(34) function.

When deciding which method to adopt for your projects, consider the following factors:

Readability: If your team values programmatic clarity, CHAR(34) offers an immediate understanding of the character being inserted.
Conciseness: If you are dealing with many nested formulas where space and character count matter, the `""""""` sequence may be slightly more compact.
Maintainability: For large spreadsheets that require future auditing or modification, the explicit nature of CHAR(34) generally leads to fewer errors than interpreting repeated double quotes.

By confidently employing either of these methods, you ensure that your strings are correctly formatted, preventing syntax errors and guaranteeing seamless data transfer to external systems. We encourage experimentation with both techniques to identify the best fit for your specific spreadsheet workflow.

Additional Resources for Google Sheets Mastery

To further expand your knowledge beyond quote escaping and optimize your data handling capabilities in Google Sheets, explore our other detailed guides covering essential functions and advanced techniques:

Mastering the CONCATENATE function for dynamic text generation.
Detailed reference guide on the CHAR and CODE functions.
Techniques for efficient drag and fill operations across large datasets.