Extracting the Last Item from Split Text in Excel: A Tutorial Using TEXTSPLIT and CHOOSECOLS

Authored by

Mohammed looti

November 10, 2025

The Evolution of Text Parsing in Excel

The capacity to efficiently dissect and reorganize textual data is arguably one of the most critical skills for any Excel power user. Data frequently enters a spreadsheet environment packaged in complex ways--be it concatenated names, intricate file paths, or long coded identifiers--all residing within a single cell. A ubiquitous task in data cleansing and preparation involves isolating the final segment of such a string, for instance, extracting the last name from a roster or retrieving the filename from a complete URL path. Historically, achieving this extraction required the painstaking nesting of traditional functions like **RIGHT**, **LEN**, and **FIND**, resulting in formulas that were often brittle and difficult to manage.

The landscape of data manipulation fundamentally changed with the introduction of modern Dynamic Array Functions in Microsoft 365. These tools offer elegant, robust, and highly readable solutions that condense complex, multi-step operations into a single, streamlined formula. Our focus here is on leveraging this new power to solve the specific challenge of retrieving the last item following a split operation. This process relies on a powerful synergy between two key functions: **TEXTSPLIT**, which efficiently breaks text into an array using a specified delimiter, and **CHOOSECOLS**, which selects precise columns from the resulting array structure.

By combining **TEXTSPLIT** and **CHOOSECOLS**, we establish a technique that dynamically adjusts to source strings of any length or complexity. This approach is superior not only because it is faster to implement, but also because it is significantly easier to audit and maintain compared to the convoluted legacy formulas. This comprehensive guide will dissect this modern technique, breaking down the essential syntax, illustrating the functional interaction, and providing a practical, step-by-step demonstration using common data scenarios to ensure you can reliably extract the final segment from any delimited text string.

Introducing the Core Dynamic Duo: TEXTSPLIT and CHOOSECOLS

The solution to reliably isolating the last segment of text lies in understanding how **TEXTSPLIT** and **CHOOSECOLS** interact within the array environment. The process begins with the TEXTSPLIT function. This function takes a text string and a defined splitting character (the delimiter) and returns the segments as elements within a horizontal array. For example, if the source text is a file path separated by backslashes, **TEXTSPLIT** transforms that single string into an array where each directory or file name occupies a separate column.

Once **TEXTSPLIT** generates this temporary array, the outer function, CHOOSECOLS, immediately takes over. Its sole purpose is to select and return specific columns from the input array. The magic occurs when we instruct **CHOOSECOLS** to count backward from the end of the array, a technique known as negative indexing. This powerful feature allows us to target the last column

position without needing to know the total number of columns generated by **TEXTSPLIT**, making the formula universally applicable.

This nested structure ensures a highly efficient workflow: the text is parsed into segments, and immediately, only the required segment (the last one) is selected and returned to the output cell. This eliminates the need for complex length calculations or searching for the final occurrence of the delimiter, simplifying data extraction dramatically.

Deconstructing the Formula: Syntax and Negative Indexing

To successfully implement this extraction method, we utilize a concise, nested formula structure. This structure must first execute the text splitting internally before the column selection can occur externally. The necessary inputs are minimal: the reference to the source cell containing the text, the specific delimiter character used to define the boundaries of the split items, and the positional index required to retrieve the final element.

The fundamental structure of the operation is shown below:

=CHOOSECOLS(TEXTSPLIT(A2, " "), -1)

Let us analyze this formula step-by-step using an example. Suppose cell **A2** contains the string "First Middle Last". The inner function, **TEXTSPLIT(A2, " ")**, executes first. It uses the space (" ") as the splitting character and returns a three-element array: {"First", "Middle", "Last"}. This temporary array is then passed as the primary argument to the outer **CHOOSECOLS** function.

The key to this entire technique is the second argument of **CHOOSECOLS**: **-1**. In Excel's modern array functions, using a negative number for column or row indices instructs the function to count backward from the end. **-1** always refers to the last column of the array, **-2** refers to the second-to-last, and so on. Therefore, by specifying **-1**, we ensure that **CHOOSECOLS** selects the final element--in this case, "Last"--regardless of how many elements **TEXTSPLIT** generated. If the original string in cell **A2** was "Chad Mike Douglas," the formula splits the string into three elements and reliably returns **Douglas**. This powerful mechanism guarantees robust data extraction across datasets where item counts per string vary widely.

Practical Application: Extracting Last Names Step-by-Step

To demonstrate the immense efficiency of this technique, we will address a common data processing requirement: isolating the last name from a column containing full names. This example is particularly relevant because names often vary in length (some may have middle initials, others full middle names, or suffixes), demanding a dynamic solution. We assume a list of full names starts in cell **A2**, as depicted in the image below.

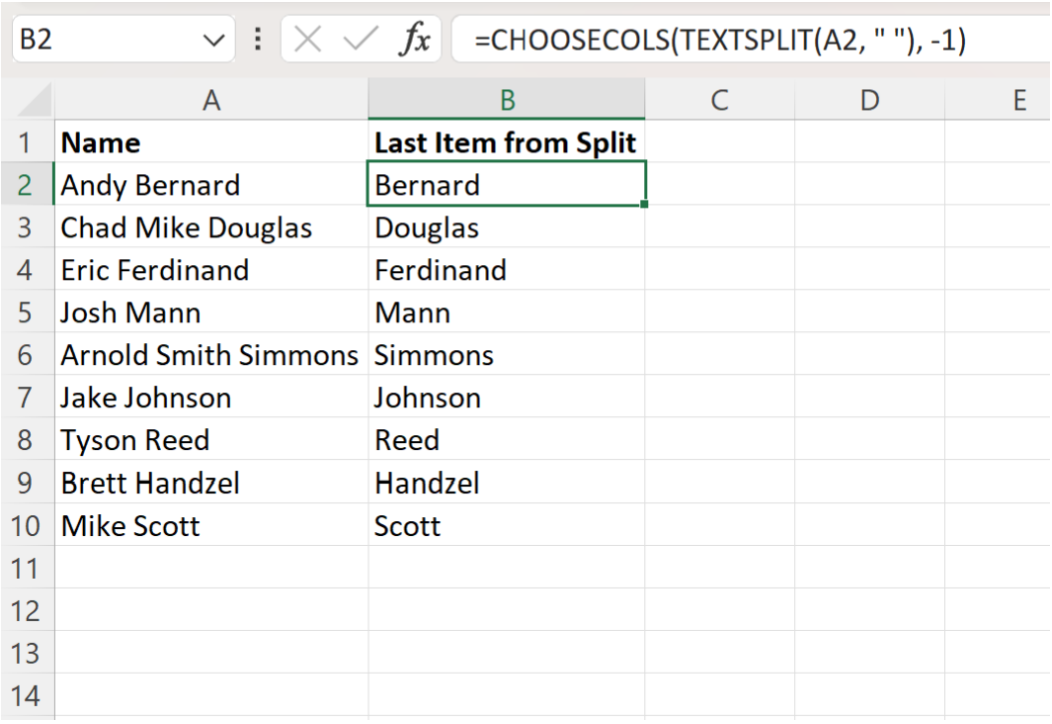
	A	B	C	D
1	**Name**			
2	Andy Bernard			
3	Chad Mike Douglas			
4	Eric Ferdinand			
5	Josh Mann			
6	Arnold Smith Simmons			
7	Jake Johnson			
8	Tyson Reed			
9	Brett Handzel			
10	Mike Scott			
11				
12				
13				
14				

Our objective is clear: split these full names using the space character as the delimiter and extract only the final segment, which corresponds to the last name. Traditional methods would require complicated logic to account for varying name lengths, but the **TEXTSPLIT** and **CHOOSECOLS** combination makes this variability irrelevant because it consistently targets the array's terminal position.

To initiate the process, simply enter the required formula into cell **B2**, the first cell in our output column. We designate cell **A2** as the text source and specify the space (" ") as the splitting character. The formula entry is:

=CHOOSECOLS(TEXTSPLIT(A2, " "), -1)

After inputting the formula in **B2**, the result immediately spills or can be copied down using the fill handle. Since the cell reference **A2** is relative, it automatically adjusts for subsequent rows (becoming **A3**, **A4**, and so on). This dynamic calculation instantly processes the last component for every name in the source column, demonstrating the significant time savings provided by modern Excel array functions when manipulating large textual datasets.



	A	B	C	D	E
1	**Name**	**Last Item from Split**			
2	Andy Bernard	Bernard			
3	Chad Mike Douglas	Douglas			
4	Eric Ferdinand	Ferdinand			
5	Josh Mann	Mann			
6	Arnold Smith Simmons	Simmons			
7	Jake Johnson	Johnson			
8	Tyson Reed	Reed			
9	Brett Handzel	Handzel			
10	Mike Scott	Scott			
11					
12					
13					
14					

As the results show, the formula successfully isolates the last segment of the split text, regardless of the complexity of the original string in column A. This robust method ensures the accurate extraction of the last name for every individual listed, relying on the guaranteed precision of **-1** indexing.

Addressing Data Quality: Handling Spaces and Legacy Methods

While the **TEXTSPLIT** and **CHOOSECOLS** combination is the most efficient solution for current Microsoft 365 users, two important practical considerations must be addressed: handling messy data and understanding the necessity of legacy methods for users on older Excel versions.

A critical aspect of data quality when using **TEXTSPLIT** is the presence of redundant spaces. If a source string contains leading, trailing, or multiple consecutive spaces (e.g., " John Doe "), **TEXTSPLIT** interprets the empty space between the two delimiters as an empty element within the resulting array. This unintended empty element can potentially shift the intended position of the "last item." To guarantee a clean split array, it is best practice to nest the **TRIM** function around the source cell reference. For example, using **TEXTSPLIT(TRIM(A2), " ")** cleans up all unnecessary spaces before the text splitting occurs, ensuring that the resulting array is perfectly structured for **CHOOSECOLS** to process.

For users operating with older versions of Excel that predate dynamic array functions (e.g., Excel 2016 or earlier), the complexity of extracting the last word is significantly higher. The traditional method involves a highly convoluted formula relying on a combination of **RIGHT**, **LEN**, **FIND**, and

often **SUBSTITUTE**. The core challenge is that the standard **FIND** function only locates the first instance of a character. To find the *last* delimiter, users must typically employ **SUBSTITUTE** to temporarily replace all spaces with a long string of unique characters (e.g., a string of 255 spaces). Then, **FIND** locates the position of the character representing the start of the final space block, and **RIGHT** and **LEN** are used to clip the text from that point forward. This legacy formula is notoriously challenging to construct, debug, and maintain, proving highly sensitive to minor data inconsistencies and demonstrating why the modern dynamic array approach is a monumental advancement.

Expanding Your Toolkit: Other Essential Dynamic Array Functions

Mastering dynamic array functions like TEXTSPLIT and CHOOSECOLS represents a paradigm shift away from traditional cell-by-cell formulas toward powerful, array-based calculations. Once comfortable with the concept of array manipulation, users can unlock a new level of efficiency in data processing and reporting within Excel.

To further enhance proficiency in text manipulation, indexing, and array management, exploring the full suite of related dynamic array functions is highly recommended. These tools offer similar flexibility and power, allowing for dynamic filtering, sorting, and aggregation with minimal effort.

Key dynamic array functions that complement text splitting and column selection include:

TEXTJOIN: The inverse of **TEXTSPLIT**, this function allows for the combination of multiple text strings or ranges using a specified delimiter.

FILTER: A function used to dynamically filter a range of data based on specific criteria, automatically spilling the resulting dataset into adjacent cells.

SORT and **SORTBY**: These functions enable the dynamic sorting of ranges or arrays based on specific columns or criteria without requiring manual intervention via Excel's built-in sorting interface.

UNIQUE: This powerful tool extracts all unique values from a specified range or array, automatically listing the distinct results.

By integrating these functions, users can create comprehensive, self-updating data models that far surpass the limitations of legacy Excel formula structures.