

Developer
/ Documentation
/ Unreal Engine ▾
/ Unreal Engine 5.4 Documentation
/ Programming and Scripting
/ Unreal Architecture
/ String Handling
/ FText

FText

Reference for creating, converting, comparing, and more with FText in Unreal Engine.



In **Unreal Engine (UE)** the primary component for [text localization](#) is the `FText` class. All user-facing text should use this class, as it supports text localization by providing the following features:

- [Creating localized text literals.](#)
- [Formatting Text](#) (to generate text from a placeholder pattern).
- [Generating text from numbers.](#)
- [Generating text from dates and times.](#)
- [Generating derived text](#), such as making text upper or lower case.


`FText` also features the `AsCultureInvariant` function (or the `INVTEXT` macro), which creates non-localized, or "culture invariant" text. This is useful for things like converting a player name from an external API into something you can display in your user interface.

You can create a blank `FText` using either `FText::GetEmpty()`, or by using just `FText()`.

Conversion

`FText` can be converted to and from `FString`. However, because `FText` contains character strings linked to localization data, while `FString` only contains a character string, these methods are inherently lossy and will discard (or fail to create) localization data. An alternative method, [Text Value Marshalling](#) provides lossless conversion, although this method produces data more suitable for internal marshalling than human consumption.

The following conversion functions produce `FText` strings, but without localization data:

<code>FText</code> Function	Description
<code>AsCultureInvariant</code>	Creates a non-localized and culture invariant <code>FText</code> instance from an existing <code>FString</code> .
<code>FromString</code>	<div>Creates a non-localized <code>FText</code> instance from an existing <code>FString</code>.</div> <div> This is identical to <code>AsCultureInvariant</code> in non-editor builds. In editor builds, this function does not flag the text as culture invariant, which means it could still become localizable if assigned to an <code>FText</code> property in a saved Asset.</div>
<code>FromName</code>	Creates a non-localized <code>FText</code> instance from an existing <code>FName</code> . This is the same as calling <code>FromString</code> on the output of the <code>FName</code> parameter's <code>ToString</code> function.

To convert from `FText` to `FString`, use the `ToString` function. The resulting `FString` will hold the string data from the `FText`, but not the localization data.

Comparison

Because `FText` data is more complex than simple strings, it does not support overloaded-operator comparisons. Instead, it provides several functions to perform comparisons that recognize the nuanced data that it contains. The following comparison functions are available:

FText	Function	Description
	EqualTo	This function takes an <code>ETextComparisonLevel</code> value to determine what comparison rules to use. It returns a <code>bool</code> indicating whether or not the calling <code>FText</code> matches the other under those comparison rules.
	EqualToCaseIgnored	This function is a wrapper for calling <code>EqualTo</code> with an <code>ETextComparisonLevel</code> value of <code>Second</code> . The return value comes directly from <code>EqualTo</code> .
	CompareTo	This function takes an <code>ETextComparisonLevel</code> value to determine what comparison rules to use. It returns an <code>int32</code> like most string- or memory-comparison functions, where zero indicates equality, and negative or positive values indicate that the calling <code>FText</code> sorts lower or higher, respectively, relative to the <code>FText</code> parameter.
	CompareToCaseIgnored	This function is a wrapper for calling <code>CompareTo</code> with an <code>ETextComparisonLevel</code> value of <code>Second</code> . The return value comes directly from <code>CompareTo</code> .

Using FText in User Interfaces

Slate/UMG

Slate and UMG use `FText` attributes or arguments for any built-in Widgets that show or manage user-facing text. We recommend using `FText` for any custom Widgets that you build.

HUD/Canvas

To display `FText` through the HUD system with Canvas, create a new `FCanvasTextItem` and set its `Text` variable to the text you want to display, as in the following example code:

```
1 // Create a new FCanvasTextItem instance to contain the text.
```

```
2 FCanvasTextItem TextItem(FVector2D::ZeroVector, TestHUDText, BigFont,  
  FLinearColor::Black);  
3 // Add the text into the FCanvasTextItem.  
4 TextItem.Text = FText::Format(LOCTEXT("ExampleFText", "You currently have {0}  
  health left."), CurrentHealth);  
5 // Draw the text to the screen with FCanvas::DrawItem.  
6 Canvas->DrawItem(TextItem, 10.0f, 10.0f);  
7
```

 Copy full snippet



When using Canvas in a HUD, you must call `DrawItem` within the `DrawHUD` function, or call it in a function chain that begins with `DrawHUD`.