String literals

Are enclosed in double quotes ("). They can be used to represent any sequence of characters, including spaces, newlines, and special characters.

Swift
```swift
let name = "John Doe"
let greeting = "Hello, \(name)!"
print(greeting) // Prints: Hello, John Doe!
```

Multiline string literals

Are enclosed in three double quotes (""""). They can be used to represent any sequence of characters, including spaces, newlines, and special characters. The newline characters are preserved in the string.

Swift
```swift
let poem = """
Roses are red,
Violets are blue,
Sugar is sweet,
And so are you.
"""
print(poem)
```

Multiline string literals are often used to represent long pieces of text, such as poems, code, or documentation. They are also useful for representing text that contains newline characters, such as email messages or HTML code.

Here is a table summarizing the differences between string literals and multiline string literals:

Feature	String literals	Multiline string literals
Enclosing characters	Double quotes (")	Three double quotes (""")
Newline characters	Are not preserved	Are preserved
Use cases	Representing short pieces of text	Representing long pieces of text, text with newline characters

Special Characters in String Literals

In Swift, special characters can be included in string literals using escape sequences. An escape sequence is a backslash (\) followed by one or more characters that represent a special character.

Here are some of the most common escape sequences in Swift:

Escape sequence	Meaning
\	Backslash
"	Double quote
'	Single quote
\n	Newline
\r	Carriage return
\t	Tab
\0	Null character
\u{n}	Unicode scalar value, where n is a 1-8 digit hexadecimal number

For example, the following code defines a string literal that contains a newline character and a tab character:

Swift

```
let message = "Hello,\n\tHow are you?"
print(message)
```

This will print the following output:

```
Hello,
      How are you?
```

You can also use escape sequences to represent characters that are not directly supported by the keyboard, such as the © copyright symbol. For example, the following code defines a string literal that contains the © copyright symbol:

Swift

```
let copyright = "© 2023 All rights reserved."
print(copyright)
```

This will print the following output:

```
© 2023 All rights reserved.
```

Escape sequences are a powerful tool for representing special characters in string literals. They make it possible to represent any character in the Unicode character set.

Here is a table summarizing the different ways to represent special characters in string literals in Swift:

Special character	Escape sequence	Unicode scalar value
Backslash	\	\u{5C}
Double quote	"	\u{22}
Single quote	'	\u{27}
Newline	\n	\u{A}
Carriage return	\r	\u{D}
Tab	\t	\u{9}
Null character	\0	\u{0}
© copyright symbol	\u{00A9}	\u{A9}

Extended String Delimiters

Extended string delimiters in Swift provide a convenient way to represent string literals containing special characters without the need for escaping. These delimiters are enclosed in either three hash symbols (###) or a single hash symbol (#).

Using Three Hash Symbols (###)

The three hash symbol delimiters are used for string literals that contain newlines or other special characters. They allow you to write multiline strings without using backslashes to escape newline characters.

Swift

```
let poem = ###"""
Roses are red,
Violets are blue,
Sugar is sweet,
And so are you.
"""###
```

Using a Single Hash Symbol (#)

The single hash symbol delimiter is used for string literals that contain special characters within interpolated expressions. It allows you to include special characters like double quotes or single quotes without escaping them.

Swift

```
let name = "John Doe"
let greeting = #"\(name), how are you?"#
print(greeting) // Prints: John Doe, how are you?
```

Benefits of Extended String Delimiters

Extended string delimiters offer several advantages over regular string literals:

- **Improved Readability:** They make string literals with special characters more readable and easier to understand.
- **Reduced Escaping:** They eliminate the need for escaping special characters, reducing the risk of typos and errors.
- **Conciseness:** They promote concise code by avoiding unnecessary backslashes and escape sequences.

Comparison with Regular String Literals

Feature	Extended String Delimiters	Regular String Literals
Enclosing Characters	Three hash symbols (###) or single hash symbol (#)	Double quotes (")
Special Character Escaping	Not required	Required
Use Cases	String literals with special characters, multiline strings	Any sequence of characters

Conclusion

Extended string delimiters are a valuable feature in Swift that enhances code readability, maintainability, and bug-free development. They simplify the handling of special characters within string literals and promote concise, expressive code.