**C# Keywords**

Keywords are words reserved by the system and have special predefined meanings when writing C# programs.

**Complete List of C# Keywords**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| abstract | as | base | bool | break |
| byte | case | catch | char | checked |
| class | const | continue | decimal | default |
| delegate | do | double | else | enum |
| event | explicit | extern | false | finally |
| fixed | float | for | foreach | goto |
| if | implicit | in | int | interface |
| internal | is | lock | long | namespace |
| new | null | object | operator | out |
| override | params | private | protected | public |
| readonly | ref | return | sbyte | sealed |
| short | sizeof | stackalloc | static | string |
| struct | switch | this | throw | true |
| try | typeof | uint | ulong | unchecked |
| unsafe | ushort | using | virtual | void |
| while |  |  |  |  |

**C# Identifiers**

Identifiers are names used to identify code elements. The class name HowdyPartner is an example of an identifier. Identifiers should be meaningful for their intended purpose. For example, the HowdyPartner program prints the words "Howdy, Partner!" to the console.

The C# character set conforms to Unicode 3.0, Technical Report 15, Annex 7. Unicode is a 16-bit character format designed to represent the many characters sets from all languages worldwide. Any Unicode character can be specified with a Unicode escape sequence, \u or \U, followed by four hex digits. For example, the Unicode escape sequence \u0043\u0023 represents the characters C#.

The decision to make the C# character set conform to Unicode standards is significant. The most prevalent character set among languages has been the American Standard Code for Information Interchange (ASCII). The primary limitation of ASCII is its 8-bit character size. This doesn't accommodate multibyte character sets for various international languages. Languages such as Java were designed with the Unicode character set built-in. As the world becomes smaller, international considerations must become larger.

Identifiers can have nearly any name, but a few restrictions apply. Here are some rules to follow when creating identifiers:

* Use nonformatting Unicode characters in any part of an identifier.
* Identifiers can begin with an allowed Unicode character or an underline.
* Begin an identifier with an @ symbol. This allows use of keywords as identifiers.

Normally, it's not permitted to use keywords as identifiers unless they're prefixed by an @ symbol. Give serious consideration before using the @ symbol because it can obfuscate code and make it confusing to read later on. There are always exceptions, but if there is a unique requirement, proceed with caution. Here are a few examples of legal C# identifiers:

currentBid

\_token

@override

\u0043sharp

Now for a few examples of invalid identifiers:

1twothree // error – 1st letter is a number

decimal // error – reserved word

\u0027format // error – Unicode formatting character

The first line is invalid because its first character is a number, which is not allowed. The first character of an identifier must be either a letter character or an underscore. The second identifier is invalid because it is a keyword. C# keywords are reserved and cannot be used as identifiers. The exception is when the keyword is prefixed with the @ character. The third line is invalid because the first character is a Unicode formatting character (#). Unicode formatting characters are not allowed in any part of an identifier.

Points to Remember :

* Keywords are reserved words that cannot be used as name or identifier.
* Prefix '@' with keywords if you want to use it as identifier.
* Identifiers must contain letters, numbers, or underscore
* Identifiers must start with @, underscore, or letter