SCALA DATA TYPES

Scala has all the same data types as Java, with the same memory footprint and precision. Following is the table giving details about all the data types available in Scala:

Data Type	Description
Byte	8 bit signed value. Range from -128 to 127
Short	16 bit signed value. Range -32768 to 32767
Int	32 bit signed value. Range -2147483648 to 2147483647
Long	64 bit signed value. -9223372036854775808 to 9223372036854775807
Float	32 bit IEEE 754 single-precision float
Double	64 bit IEEE 754 double-precision float
Char	16 bit unsigned Unicode character. Range from U+0000 to U+FFFF
String	A sequence of Chars
Boolean	Either the literal true or the literal false
Unit	Corresponds to no value
Null	null or empty reference
Nothing	The subtype of every other type; includes no values
Any	The supertype of any type; any object is of type *Any*
AnyRef	The supertype of any reference type

All the data types listed above are objects. There are no primitive types like in Java. This means that you can call methods on an Int, Long, etc.

Scala Basic Literals:

The rules Scala uses for literals are simple and intuitive. This section explains all basic Scala Literals.

Integer Literals

Integer literals are usually of type Int, or of type Long when followed by a L or l suffix. Here are some integer literals:

```
0
035
21
0xFFFFFFFF
0777L
```

Floating Point Literals

Floating point literals are of type Float when followed by a floating point type suffix F or f, and are of type Double otherwise. Here are some floating point literals:

```
0.0
```

```
1e30f
3.14159f
1.0e100
.1
```

Boolean Literals

The boolean literals **true** and **false** are members of type Boolean.

Symbol Literals

A symbol literal 'x is a shorthand for the expression **scala.Symbol** " x " . Symbol is a case class, which is defined as follows.

```
package scala
final case class Symbol private (name: String) {
   override def toString: String = "'" + name
}
```

Character Literals

A character literal is a single character enclosed in quotes. The character is either a printable unicode character or is described by an escape sequence. Here are some character literals:

```
'a'
'\u0041'
'\n'
'\t'
```

String Literals

A string literal is a sequence of characters in double quotes. The characters are either printable unicode character or are described by escape sequences. Here are some string literals:

```
"Hello,\nWorld!"
"This string contains a \" character."
```

Multi-Line Strings

A multi-line string literal is a sequence of characters enclosed in triple quotes """ ... """. The sequence of characters is arbitrary, except that it may contain three or more consuctive quote characters only at the very end.

Characters must not necessarily be printable; newlines or other control characters are also permitted. Here is a multi-line string literal:

```
"""the present string
spans three
lines."""
```

The Null Value

The null value is of type **scala.Null** and is thus compatible with every reference type. It denotes a reference value which refers to a special "null" object.

Escape Sequences:

The following escape sequences are recognized in character and string literals.

Escape Sequences	Unicode	Description

\b	\u0008	backspace BS
\t	\u0009	horizontal tab HT
\n	\u000c	formfeed FF
\f	\u000c	formfeed FF
\r	\u000d	carriage return CR
\"	\u0022	double quote "
\'	\u0027	single quote .
\\	\u005c	backslash \

A character with Unicode between 0 and 255 may also be represented by an octal escape, i.e., a backslash '\' followed by a sequence of up to three octal characters. Following is the example to show few escape sequence characters:

```
object Test {
   def main(args: Array[String]) {
      println("Hello\tWorld\n\n" );
   }
}
```

When the above code is compiled and executed, it produces the following result:

```
Hello    World
```