# Why Parentheses Are Useful in LiveCode

*by Richard Gaskin*
[Fourth World Systems]

LiveCode strives to be as English-like as practical, but not more so. English is a funny language, an odd mish-mash of Greek, Latin, Frech, German, and other roots, and is wrought with ambiguity. What would make sense to any reasonably intelligent English speaker can sometimes require too much contextual awareness for a machine too stupid to count past 1.
Concatenating expressions in LiveCode are sometimes those circumstances, especially when the expressions are used as object references.
Parentheses are not always needed by the LiveCode compiler, but because they sometimes are it can be helpful to use them for concatenation phrases, and it tends to make the code more readable as well.

## For the Compiler

Consider this statement:

```
put "Hello World" into fld "test"&1
```

If you run that in the Message Box, even if you have a field named "test1", the engine will complain that it can't find the object.
Why?
Because the engine will generally work on a statement from left to right, unless it find any parenthetical expression, in which case it will evaluate those first.
So the steps it would have to take with that example are:
1. Find field "test"
2. Put "Hello World" into it
3. Try to figure out what to do with "&1"

In fact, the engine won't even be able to execute it at all, complaining with a compiltion error when you try to set the script, since "&1" isn't a valid command or function. In this case it won't even run the first two steps, because the error will prevent compilation from occurring at all.
We could at least move past the compilation error if we used parentheses on the second object reference the compiler complained to us about:

```
put fld "test"&2 into fld ("test"&1)
```

In that case, the engine's steps would be:

1. Evaluate the parenthetical expression to arrive at "test1"
2. Find a field named "test"
3. Get the contents of field "test"
4. Put 2 after that field's value
5. Find a field named "test1"
6. Put the concatenated value into that field

That is, if the statement could be successfully executed.
The engine will be able to compile the statement because it makes syntactic sense, but at runtime will throw an execution error on step #2 because there is no field "test".
But if you used parentheses on both, like this:

```
put fld ("test"&2) into fld ("test"&1)
```

...then the engine will do this:

1. Evaluate the first parenthetical expression to arrive at "test2"
2. Evaluate the second parenthetical expression to arrive at "test1"
3. Find a field named "test2"
4. Get its contents
5. Find a field named "test1"
6. Put the retrieved contents into that field

Another example that sometimes trips up newcomers to LiveCode is when using URL syntax to read files. At first glance this might seem to work:

```
put url "file:"& tMyFilePath into fld "test1"
```

But to the engine, it sees the steps required to fulfill that request as:

1. Find the file "file:"
2. Read its contents into memory
3. Append those contents with the contents of the variable tMyFilePath
4. Find a field named "test1"
5. Put the combined value into that field

Here we don't even get out of the starting gate, with execution failing on step #1.
Once again, parentheses make it clear to LiveCode that tMyFilePath is part of the file specifier:

```
put url ("file:"& tMyFilePath) into fld "test1"
```

...then the engine will do this:

1. Evaluate the parenthecal expression to form a complete file reference
2. Find the specified file
3. Read its contents into memory
4. Find a field named "test1"
5. Put the file contents into that field

So while there are cases where the engine won't complain about not using parentheses, it's a good habit to use them consistently for the many cases where they may indeed make a substantial difference.

## For the Developer

The other benefit using parentheses is readability. Or more specifically, skimmability.

Code is rarely read per se. It's not a novel, and only the most ardent LiveCode fan would curl up next to the fire with a brandy in their hand enjoying a printout of LiveCode scripts as others might with Chaucer or Hemmingway.

We usually only read code when we're trying to fix it or enhance it. In those cases we're not savoring every word as we would with our favorite novels, but just skimming it for the relevant portions that will help us identify the problem at hand so we can fix it and move on to more interesting things.

When concatenating expressions in LiveCode, parentheses help put a visual boundary around the parts that will ultimately form a single string. This is not only sometimes useful for the compiler as we've seen above, but also useful for the human eye as we scan page after page of code looking for the elusive bug that brought us to open the script editor in the first place.

Squint your eyes a bit, so that your vision will be almost as impaired as it might at the end of a long day of debugging, and look at this line:

```
put fld "test2"&"Hello World, it's"&&the time&"!" into
fld "test1"
```

It's perfectly valid code as far as the compiler is concerned, but compare how much easier it is to quickly identify the portions that will form a single string if you add parentheses:

```
put (fld "test2" & "Hello World, it's" && the time &
"!") into fld "test1"
```

I've also taken the liberty of adding some breathing room with white space around the concatenation operator "&" to make each of the distinct elements that much clearer.

The engine doesn't care whether the code is compact or not, but your tired eyes will thank you months from now when you decide to go back to add more features.

When parentheses are required by the LiveCode engine, it'll let you know. And even when not striclty required, the benefits for human skimmability make liberal use of them a helpful habit.