An Investigation into how R handles reserved keywords

I decided to investigate how R handles reserved keywords in the language. Different languages have different philosophies about whether the reserved keywords should themselves be standard functions or not. R sort of does a mish-mash of stuff.

```
typeof(base::stop)
```

```
## [1] "closure"
```

This is the standard type for unevaluated functions in an environment. Maybe R reserved keywords are just normal functions/closures?

```
typeof(base::if)
# Error: unexpected 'if' in "typeof(base::if"
```

It thinks you're trying to put a command in the wrong place and errors. Maybe if I back-tick it it'll work.

```
typeof(base::`if`) # type "special"
```

```
## [1] "special"
```

Yep.

```
typeof(base::`+`)
```

```
## [1] "builtin"
```

```
typeof(base::any)
```

```
## [1] "builtin"
```

Wait, didn't you just tell me it was "special"? So I guess I need to backtick any item that's a non-alphanumeric symbol (i.e. the name is itself special) but among alphanumeric symbols, only type "special" need to be backticked.

What makes a special function special?

```
class(base::`if`) # function
```

```
## [1] "function"
```

```
mode(base::`if`) # function
```

```
## [1] "function"
```

Only its type.

```
base::`if` # shows source, such as it is
```

```
## .Primitive("if")
```

Again, standard source display, as long as we backtick.

R-lang tells me that special is for "internal functions that do not evaluate their arguments" and "builtin" is for "internal functions that do evaluate their arguments". Okay, fair. "builtin" seems like a superfluous type; there's no reason this stuff can't be closures. But "special" actually matters. R-internals tells me a few of the "special" primitives and why they are special primitives.

But wait, R-internals claims that "seq.int" and "rep" are special type?

```
typeof(seq.int)
```

```
## [1] "builtin"
```

```
typeof(rep)
```

```
## [1] "special"
```

```
base::rep
```

```
## function (x, ...)  .Primitive("rep")
```

```
base::seq.int
```

```
## function (from, to, by, length.out, along.with, ...)  .Primitive("seq.int")
```

Okay, so seq.int isn't special, notwithstanding what the manual just told me, and rep is special but I can still view its source without backticks. So there's some subset of special types that are both special AND language reserved keywords that need to be backticked?

Let's figure out if I can find out exactly which those are. This `builtins()` function is meant to be a list of all R's built-in objects:

```
bi = builtins()
head(bi)
```

```
## [1] "zapsmall"              "xzfile"                "xtfrm.Surv"
## [4] "xtfrm.POSIXlt"         "xtfrm.POSIXct"         "xtfrm.numeric_version"
```

But many of these objects are just objects in built-in packages, nothing to do with the builtin or special types. Let's try to figure out which of these are random shit, and which are actually "special". I can't remember which NSE I need in base R to do this so I'll use rlang:

```
library(rlang)
type_list_bis = sapply(builtins(), function(x) { typeof(eval(parse_expr(paste0("`", x, "`")))) })
table(type_list_bis)
```

```
## type_list_bis
##     builtin   character     closure      double environment        list
##         161          11        1084           2           8          14
##     logical        NULL    pairlist     special
##           3           2           2          37
```

I assume the NULL includes the actual backtick operator, maybe? We're getting a little closer, though. Let's subset to those that are "special" type and then grepl for those we wouldn't otherwise backtick – i.e. it's obvious we'd backtick the `[[` operator or whatever.

```
candidates = names(type_list_bis)[which(type_list_bis == "special")]
candidates[grepl("^[A-Za-z_]*$", candidates)]
```

```
##  [1] "while"        "UseMethod"    "switch"       "substitute"
##  [5] "signif"       "round"        "return"       "repeat"
##  [9] "rep"          "quote"        "next"         "missing"
## [13] "log"          "if"           "function"     "forceAndCall"
## [17] "for"          "expression"   "call"         "break"
```

We're down to 20 plausible special types. Which need to be backticked? Let's do this the stupid way of just trying to print the source without backticking, then if it fails, keep it. There are exactly 7 actual reserved keywords in R:

```
needs_backtick = unname(unlist(sapply(candidates[grepl("^[A-Za-z_]*$", candidates)],
                        function(x) {
                          tryCatch({
```

```
                            eval(parse_expr(paste0("base::", x)))
                            NULL
                          }, error = function(e) {
                            return(x)
                          })
                        })))

needs_backtick
```

```
## [1] "while"    "repeat"   "next"     "if"       "function" "for"
## [7] "break"
```

This makes sense, it's all the very basic stuff needed for flow control. And indeed if we run `?while`, we get the R documentation for "Reserved words in R". The reserved words in R's parser are: "if", "else", "repeat", "while", "function", "for", "in", "next", "break", and then the basic types and numbers.

Two interesting things here:

1. Which items are in the reserved word list but NOT in our list above? "else" and "in". Let's see what their deal is.

```
base::`else`
# Error in get(name, envir = ns, inherits = FALSE) : object 'else' not found
```

Wait, what? "else" isn't actually an object anywhere? Nope, apparently not. It's just a magic reserved word that doesn't exist anywhere in the namespace. Enjoy. "in" as well. They work just like magic. You can't check their types because they don't exist.

2. Switch is the poor neglected cousin of the default operators:

```
typeof(base::switch)
```

```
## [1] "special"
```

```
base::switch
```

```
## function (EXPR, ...)  .Primitive("switch")
```

In summary:

- Most built-in objects are just normal objects
- Some are type "builtin", but this is totally irrelevant
- Some are type "special", but this is still totally irrelevant
- A subset of the "special" types are reserved words, which must be backticked. These are mostly flow control operators.
- Some flow control operators are not reserved words.
- There are also some reserved words which don't appear to be objects at all.