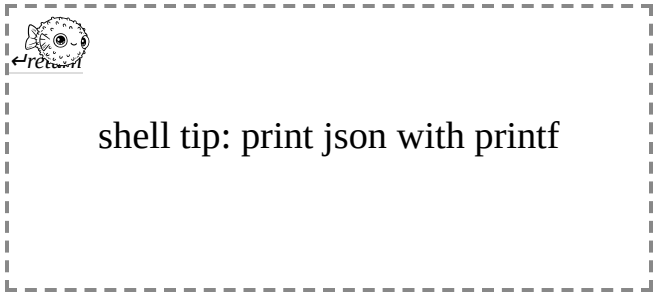

shell tip: print json with printf
2023-10-06

i am commonly consulted when people are having strange issues with shell.

one of the most common issues that shows up is when people try printing their
json in shell.

printing json might seem straightforward at first, just:

```
$ echo '{"name": "jes", "sign": "aquarius"}'
{"name": "jes", "sign": "aquarius"}
```

but inevitably, variables show up:

```
$ echo '{"name": "$name", "sign": "$sign"}'
{"name": "$name", "sign": "$sign"} <-- this output is very broken
```

at this point, the shell user realizes that they need double quotes in order to make variables interpolate properly.

the user then commits one of two *cardinal* sins:

sin 1: the eternal double-quote swamp

```
$ echo "{\"name\": \"$name\", \"sign\": \"aquarius\"}"
{"name": "jes", "sign": "aquarius"}
```

if you write anything as ugly as the abomination above in shell, you are sending a clear signal that you do not care about yourself. you are condemning
yourself to wade through the double-quote swamp for eternity.
and you will deserve it.

```
'\"\"\"\"\"\"\"\"\"\"\"'\\"\"\"\"\"\"\\"\"\"\"\\\"\"\"\"\"\"\"\"\\"\"\"
   ^ a small portion of the eternal double-quote swamp ^
```

sin 2: the long ugly HEREDOC

```
$ cat <<-EOF
```

```
    {"name": "$name", "sign": "$sign"}
    EOF
    {"name": "jes", "sign": "aquarius"}
```

the HEREDOC is more forgivable, but it takes up 2 unnecessary lines, and it's
much harder to recall on-the-fly. and frankly, it looks pretty strange.

and god forbid if you have to use a heredoc in an indented block.

kiss your nice orderly indentation goodbye:

```
    if blah; then
        cat <<-EOF
    {"name": "$name", "sign": "$sign"}
    EOF
    fi
```

there is a better way! we just need to bust out our old dusty friend *printf*

printf offers several advantages over echo:
- many languages (C, Go) have a printf equivalent, so it feels familiar
- printf behavior does not differ across systems (it is posix compliant)
- printf handles escape sequences (printf "hello\nworld\n")
- printf can trivially handle json

witness, as we print the same json text using printf:

```
    $ printf '{"name": "%s", "sign": "%s"}' "$name" "$sign"
    {"name": "jes", "sign": "aquarius"}
```

& there you have it! nice orderly json using an easy to remember command, with
no-fuss variable injection built-in.

i hope that this tip saves you some pain!

love,
jes