

# Speaker Notes: Ammonite: Succinct Scala Shell Scripting

---

Markus Dale, [medale@asymmetrik.com](mailto:medale@asymmetrik.com)

October 2019

- Open Ammonite site <https://ammonite.io>
- Download page: <https://github.com/lihaoyi/Ammonite/releases>
- Requests lib: <https://github.com/lihaoyi/requests-scala>
- CSV data: [https://catalog.data.gov/dataset?res\\_format=CSV&page=2](https://catalog.data.gov/dataset?res_format=CSV&page=2),  
<https://data.ny.gov/api/views/e8ky-4vqe/rows.csv?accessType=DOWNLOAD>

- Bio:
  - mostly Java, big data with Hadoop
  - big data with Spark, Databricks, Scala
  - Now Asymmetrik - Scala, Spark, Elasticsearch, Akka...
  - Data Engineer
  - Slides: <https://github.com/medale/prez-ammonite-scala-shell/blob/master/presentation/SparkDataEngineering.pdf>
  - Code Examples:  
<https://github.com/medale/prez-ammonite-scala-shell/code>

## Examples of Shell Scripting

- Automate a small but labor-intensive task
- Run clean-up jobs with cron
- Refresh all git repositories (git pull)
- build and publish all projects under a common dir

- hard to remember if not used all the time
- each command has its own set of options
- Can pipe things together - very useful
- No types - mostly treated as strings (other than exit code)
- Dr. Google - but fragile to maintain

- Use java import
- More verbose

- from 1 liner to 8 (and don't forget close)

- Domain-specific language for Scala shell scripting
- Make common tasks as terse as possible (but typed)
- imports Scala commands and implicits to convert strings to paths
- spawn subprocesses and pipes
- additional ease of use to remove need for full project/build system



- Open <https://github.com/lihaoyi/Ammonite/releases>

## Other ways of installing Ammonite

Configure the shell `~/ammonite/predef.sc`

## Major improvements over Scala REPL

- open Scala REPL and Ammonite
- Syntax highlighting input/output
- output valid Scala code: `Seq.fill(10)("Hello, Ammonite")`
- multi-line editing (up arrow and search)
- Classpath search: `StandardCharsets<TAB>` (if not imported yet but on classpath)

## Magic import \$file: Importing scripts

```
import $file.CommonImports
CommonImports.printCwd //accesses Paths OK
Paths.get("foo") //no imports - error
CommonImports.printWithNewlines(List(1,2,3))
```

```
import $file.scripts.Utills
Utills.du()
import Utills._
du()
```

```
import $file.^..print
print.message()
```

## Magic import \$exec: Bring in the defs AND imports

- all definitions AND import statements

```
//contains: import java.nio.file._
```

```
import $exec.CommonImports  
Paths.get("foo")  
import CommonImports._  
printlnWithNewlines(List(1,2,3))
```

## Magic import \$ivy: Download libraries and their dependencies

```
import $ivy.`com.univocity:<TAB>`
```

```
import $ivy.`com.univocity:univocity-parsers:<TAB>`
```

```
import $ivy.`com.univocity:univocity-parsers:2.8.3`
```

```
CsvParser<TAB>
```

```
import com.univocity.parsers.csv.CsvParser
```

```
...
```

```
repl.imports      //user imports only  
repl.fullImports  //also Ammonite imports  
  
repl.clipboard.read/write  
  
repl.clipboard.write("Hello from Ammonite")
```



```
interp.load.cp('lib/"baz.jar")
```

```
import foo.bar.Baz
```

```
Baz.VeryImportantNumber
```

```
source(Baz)
```

```
import java.net.URL
```

```
source(URL) //no - has to be object
```

```
source(new URL("http://foo.bar"))
```

```
Path("/tmp")
```

```
RelPath("lib")
```

```
Path("temp")
```

```
RelPath("/lib")
```

```
val dir = "/tmp/rest"
```

```
ls! dir //no
```

```
Path("/tmp/rest")
```

```
root/'tmp/'rest
```

## Special path variables

- home - `$HOME` or `user.home`
- root - `ls!` `root == Bash: ls /`
- pwd - `sys.props("user.dir")` //does not change
- wd - changes with `cd!` command
- up - `..` in path, e.g. `root/'tmp/up == root`

- `ls! == ls! wd == ls(wd)`
- `rm! dir - Bash: rm -R dir`
- `cp(src,dest)`
- `mv(src,dest)`
- `mkdir! newDirPath - Bash: mkdir -p newDir`
- `stat! filePath`

## Pipes and grep!

```
def sum(a: Int, b: Int) = a + b
```

```
List(1,2,3) |& sum
```

```
ls! wd |? grep! "foo|bar".r
```

## Built-in libraries: upickle and requests-scala

```
val lines =  
  read.lines('input / "ids.json", StandardCharsets.UTF_8)  
val jsons = lines.map { l =>  
  ujson.read(l)  
}  
jsons.foreach { obj =>  
  obj("source") = ujson.Str("asymmetrik")  
  obj("dest") = ujson.Num(42)  
}  
jsons.foreach { obj =>  
  obj.obj.remove("dest")  
}  
val response = requests.post("http://httpbin.org/post",  
  headers = Map("accept" -> "application/json"),  
  data = ujson.write(ujson.Obj("fp" -> "map")))
```

- WikipediaVocab.sc
- Spreadsheet.sc