Edit

Makefile cheatsheet

```
uglify = $(uglify)        # lazy assignment
compressor := $(uglify)   # immediate assignment
prefix ?= /usr/local      # safe assignment
```

= expressions are only evaluated when they're being used.

Magic variables

```
out.o: src.c src.h
  $@   # "out.o" (target)
  $<   # "src.c" (first prerequisite)
  $^   # "src.c src.h" (all prerequisites)

%.o: %.c
  $*   # the 'stem' with which an implicit rule matches ("foo" in "foo.c")

also:
  $+   # prerequisites (all, with duplication)
  $?   # prerequisites (new ones)
  $|   # prerequisites (order-only?)

  $(@D) # target directory
```

Command prefixes

`-`	Ignore errors
`@`	Don't print command

`+`	Run even if Make is in ‘don’t execute’ mode

```
build:
    @echo "compiling"
    -gcc $< $@

-include .depend
```

Find files

```
js_files  := $(wildcard test/*.js)
all_files := $(shell find images -name "*")
```

Substitutions

```
file     = $(SOURCE:.cpp=.o)   # foo.cpp => foo.o
outputs  = $(files:src/%.coffee=lib/%.js)

outputs  = $(patsubst %.c, %.o, $(wildcard *.c))
assets   = $(patsubst images/%, assets/%, $(wildcard images/*))
```

More functions

```
$(strip $(string_var))

$(filter %.less, $(files))
$(filter-out %.less, $(files))
```

Building files

```
%.o: %.c
  ffmpeg -i $< > $@   # Input and output
  foo $^
```

Includes

```
-include foo.make
```

Options

```
make
  -e, --environment-overrides
  -B, --always-make
  -s, --silent

  -j, --jobs=N   # parallel processing
```

Conditionals

```
foo: $(objects)
ifeq ($(CC),gcc)
        $(CC) -o foo $(objects) $(libs_for_gcc)
else
        $(CC) -o foo $(objects) $(normal_libs)
endif
```

Recursive

```
deploy:
  $(MAKE) deploy2
```

Further reading

- isaacs's Makefile
- Your Makefiles are wrong
- Manual