Kenny Gao, Mike Lester, Eric Reed

- Based on Forth (Team “May the Forth Be With You”)
- Created in 2003 by ~~Slava Pestov~~ a genius
- Stack-based
- Concatenative
- Currently at version 0.94 (and in constant development)

Q1

Stack Programming Basics

- Arguments are pushed onto the stack implicitly
- Stack is used to pass arguments and results around
- Operations modify the stack
  - *Stack effects* describe the changes that occur
- notation Postfix !

Concatenative Programming Basics

- Everything is a function
- Juxtaposition defines function composition

  ```
  a b = a ∘ b
  ```

- `load-image process-image display-image`

Getting Started with Factor

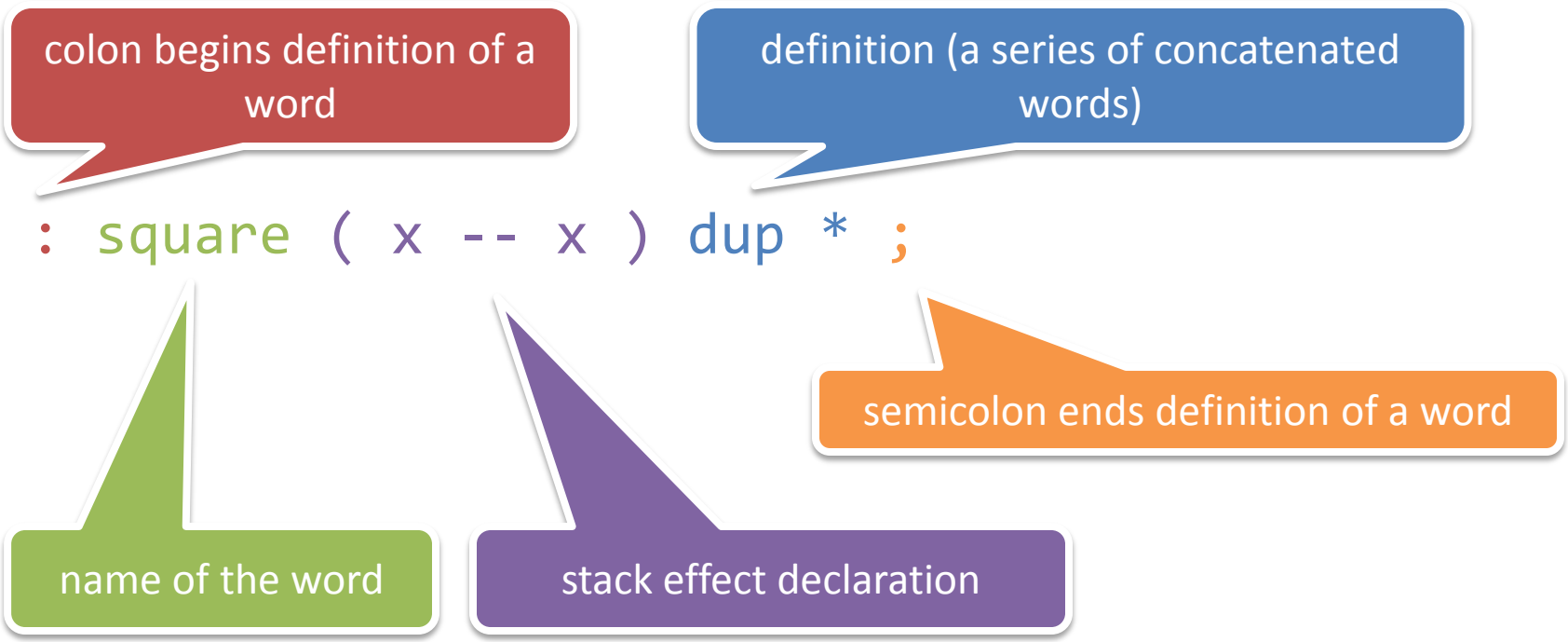
- Functions in Factor are called *words*
  - Typically very short and concise
- Modules in Factor are called *vocabularies*
  - Only used for namespacing and organization
  - Think Java packages
- Words are defined from other words
  - primitives = base case

Q2

Examples

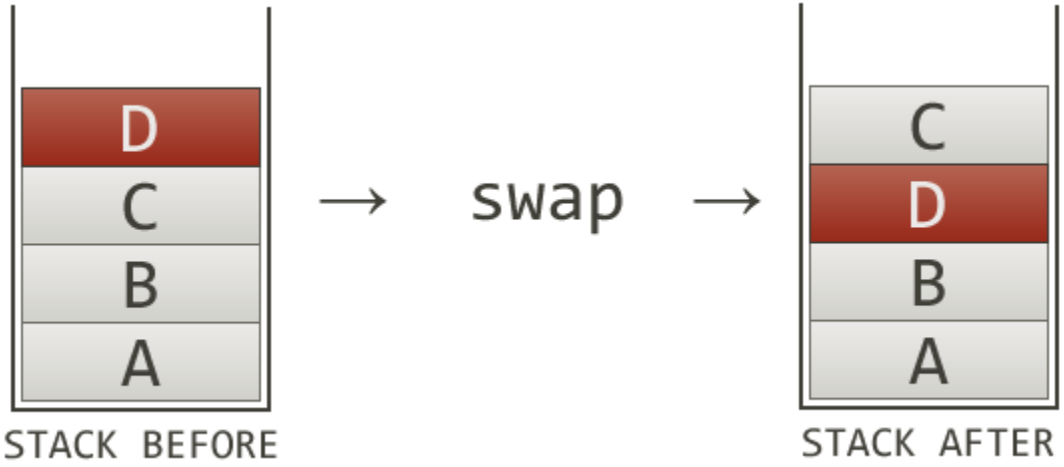
- 3 .
- "hello world" .
- 6 7 *
- 3 +
- drop
- 10 sq 5 - .

Anatomy of a Word



Stack Effect Declarations

- Exactly what it sounds like !
- Example !
  - `swap ( x y -- y x )`



http://elasticdog.com/2008/12/beginning-factor-shufflers-and-combinators/

Q4, 5

Quotations

- Quotations are bits of code pushed onto the stack for delayed execution
- Like LISP/Scheme quotations!
- Form: [code later to run]
- You can nest quotations too
- Useful for higher-order words
- Code as data! You can build up quotations dynamically (again like LISP)

Combinators

- A word that takes code as input
- Examples (top of the stack is on the right):

```
3 5 [ 1 + ] dip
```

dip applies a quotation to the second thing on the stack, ignoring the top

```
{ 1 2 3 } [ sum ] [ length ] bi /
```

bi applies two quotations to the same value and places both results on the stack. Here we use it for a mean operation. >:[

```
3 10 < [ “Math OK” print ] [ “Math FUBAR” print ] if
```

if takes a boolean, a quotation for the true case, and a quotation for the false case.