namespaces & variables

Daniel Jackson
What's in a name? That which we call a rose
By any other name would smell as sweet.

—Shakespeare; Romeo & Juliet, 2:2
namespaces

context matters
› same name, different meaning

applications of this idea
› program elements
› state components
› files & directories
› URLs & routing
› ...

![Diagram](image)
environments

environment
› namespace for program variables

in Javascript
› every bound variable has a value
› value may be “undefined”

confusing
› unbound var gives ref error
› property can only be undefined
› undefined is a value!
lookup

to evaluate an expression
› lookup value of each var
› apply functions to arguments

how to lookup
› just find the binding for the var
assignment

assignment statement
› x = e, read “x gets e”

semantics
› evaluate e to value v
› if x is bound, replace value with v
› else create new binding of x to v

in JS, all names are vars
› function names can be reassigned
› can define “undefined” (!)

contrast to Java
› variables just one kind of name
› method, class, package names; also constants like “null”

> h = "hello there"
> "hello there"
> escape(h)
> "hello%20there"
> escape = function()
> {return "gone!";}
function () {return "gone!";}
> escape(h)
> "gone!"

> var foo
undefined
> foo === undefined
true
> undefined = 3
3
> foo === undefined
false
objects as environments

what is $x.f = e$?
  › can view as an assignment?
  › not of the name “$x.f$”, since fails if $x$ is unbound

object as environment
  › $x$ is itself a mini-environment
  › $x.f = e$ binds $f$ in the environment called $x$

not quite right...
  › a bit of non-uniformity
  › unbound variable gives reference error
  › property of object undefined until set

> x = {}
Object
> x.f = 1
1
> x.f
1
> x.g
undefined
aliasing

after the assignment \( x = y \)
\( \Rightarrow \) \( x \) is bound to same value as \( y \)

how sharing arises
\( \Rightarrow \) no implicit copying
\( \Rightarrow \) so \( x \) and \( y \) are names for same object

consequence
\( \Rightarrow \) change to “one” affects the “other”

if object is immutable
\( \Rightarrow \) no change to object possible
\( \Rightarrow \) so as if value is copied

```plaintext
> y = []
[ ]
> x = y
[]
> x.f = 1
1
> y.f
1
```
evaluating expressions

suppose you see an expression e
  › eg, e is f()
  › what might expression do?

evaluation can have 3 effects
  › value is returned (or exception thrown)
  › objects are modified
  › environment is updated

a puzzle
  › declare f so that f()===f() evals to false
solution to puzzle

one of several possible

```javascript
f = function () {
  f = function () {
    return 1;
  };
  return 2;
};
```