namespaces & variables

Daniel Jackson
namespaces

context matters
› same name, different meaning

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

Namespace

<table>
<thead>
<tr>
<th>Name</th>
<th>Thing</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="image" /></td>
<td><img src="image2.png" alt="image" /></td>
</tr>
</tbody>
</table>
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

> h = "hello there"
"hello there"
> escape
function escape()
{ [native code] }
> escape(h)
"hello%20there"
 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
 › a function name is just a var, can reassign
 › more on this when we see recursion

 contrast to Java
 › variables just one kind of name
 › other kinds of name: methods, classes, packages

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

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

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

consequence
  change to “one” affects the “other”

if object is immutable
  no change to object possible
  so as if value is copied