CS 5142
Scripting Languages

10/30/2013

Prelim 2 Review

Variable Declarations

Implicit	`print $a + 1;` `$b = 5;`	Read **`undef`** if non-existent
Local, lexical scope	`my $c;` `my ($d,$e)=(3,4);`	

Global used locally	`sub f{`
	`  our $g;`
	`  print $g++`
	`}`
		Hides locals; unlimited lifetime

Local, dynamic scope	`sub h{print $i;}`
	`sub k{`
	`  local $i=5; h`
	`}`
		Can also localize single array/hash item

Concepts

Static vs. Dynamic Scoping

Static scoping	Dynamic scoping
Bound in closest nesting scope in program text	Bound in closest calling function at runtime

Static scoping:

```
#!/usr/bin/perl -w
$x = 's';
sub g {
  my $x = 'd';
  return h()
}
sub h {
  return $x
}
print g(), "\n"; #s
print $x,  "\n"; #s
```

Dynamic scoping:

```
#!/usr/bin/perl -w
$x = 's';
sub g {
  local $x = 'd';
  return h()
}
sub h {
  return $x
}
print g(), "\n"; #d
print $x,  "\n"; #s
```

Example of `m//` Matching

Input	`-rw-r--r--   1 soule  soule     3998 Aug 29 12:49 hw01.html`
	`-rw-r--r--@  1 soule  soule     4480 Sep  5 14:10 hw02.html`
	`-rw-------   1 soule  soule     8868 Sep  9 12:09 index.html`

Script

```
#!/usr/bin/perl -w
while($line = <>) {
  if ($line =~ m/(\d+)\s\w{3}[\d\s:]+(\S+)$/) {
    printf "$2 contains $1 bytes\n";
  }
}
```

Output

```
hw01.html contains 3998 bytes
hw02.html contains 4480 bytes
index.html contains 8868 bytes
```

Perl

Inside a Regular Expression

Kind	Construct	Syntax	Example: Pattern	Example: Matches
Essentials	Character	Itself	`b`	`b`
	Concatenation	$e_1 e_2$	`bc`	`bc`
	Alternative (or)	e_1 `|` e_2	`a|bc`	`a`, `bc`
	Repetition (≥0)	e`*`	`a*`	, `a`, `aa`, `aaa`
	Grouping	`(`e`)`	`(a|b)c`	`ac`, `bc`
Quantifier	Optional	e`?`	`(a|b)?`	, `a`, `b`
	Repetition (≥1)	e`+`	`a+`	`a`, `aa`, `aaa`
	Repetition	e`{`n`}`	`a{3}`	`aaa`
Char class	Custom class	`[…]`	`[a-c]`	`a`, `b`, `c`
	Wildcard character	`.`	`.`	`a`, `3`, `:`
	Shortcut class	`\…`	`\d`	`0`,`1`,`2`,…,`9`
Assertion	Zero-width anchor	`$`,`^`,`\b`,…	`\b`	(word boundary)

Non-Regular Perl Regex Features

- Non-regular = not essential feature, and can not be emulated by essential features
  - Backtracking engine, may take exponential time
- Backreferences in same pattern: `\1`, `\2`, …
- Zero-width assertions:
  - Look-ahead positive `(?=…)`, negative `(?!…)`
  - Look-behind positive `(?<=…)`, negative `(?<!…)`
- Match-time code execution: `(?{…})`
- Match-time interpolation: `(??{…})`
- Backtracking suppression: `(?>…)`

Non-Regular Perl Regex Examples

Description	Regex	Matched
Square (repeated)	`(\d+)x\1`	`123x123`
Palindrome (mirrored)	`(\w+)\w(??{reverse $1})`	`redivider`
Balanced parentheses	`(<+)x(??{'>' x length $1})`	`<<<x>>>`

•PHP

Anonymous Functions

- Creating new variable function from strings:

```
$x=create_function(args, code);
```

- Example script:

```
<?php
function callFn($fn, $x) { $fn($x); }
function printIt($it) { print "printIt $it\n"; }
#pass reference to named subroutine; prints "printIt Hello"
callFn("printIt", "Hello");
#pass reference to anonymous subroutine; prints "lambda Hi"
callFn(create_function('$it', 'print "lambda $it\n";'), "Hi");
  ?>
```

- Useful for call-backs, e.g., `usort(array, cmp_function)`

Inheritance in JavaScript

```
function Fruit(weight) {
  this.weight = weight;
}
Fruit.prototype.pluck = function() {
  return "fruit(" + this.weight + "g)";
}
Fruit.prototype.prepare = function(how) {
  return how + "d " + this.pluck();
}
function Apple(weight, color) {
  this.weight = weight;
  this.color = color;
}
Apple.prototype = new Fruit();
delete Apple.prototype.weight;
Apple.prototype.constructor = Apple;
Apple.prototype.pluck = function() {
  return this.color + " apple";
}
```

- Constructor assigns fields
- Top-level assigns methods
- Inherit from Fruit.prototype
- Remove spurious property
- Enable instance checks

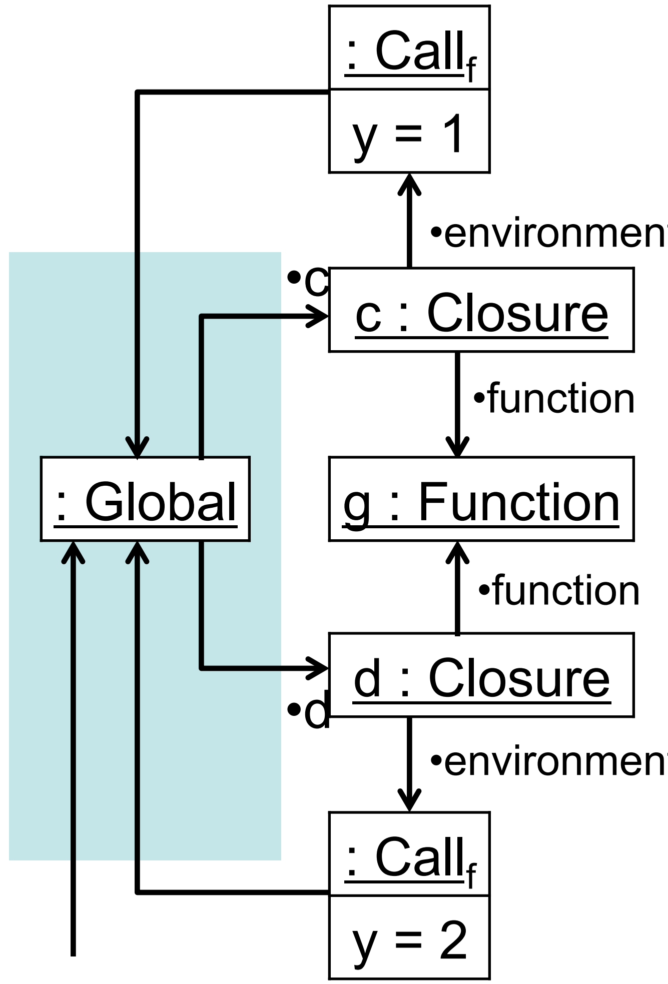
Operators on Objects

new C **(...)**	• Create new empty object o • Set o **.prototype =** C **.prototype** • Set o **.constructor** **=** C **.prototype.constructor** • Call C **(...)**, pass o as value for **this** • Return result of C **(...)** or o if none
o **.** p	• If object o has property p, return it • Otherwise, look in o **.prototype**
o **.** p **=** *expr*	• If object o has property p, assign it • Otherwise, create it and assign it
o **instanceof** C	• If o **.constructor** is C, return **true** • Otherwise, look in o **.prototype**

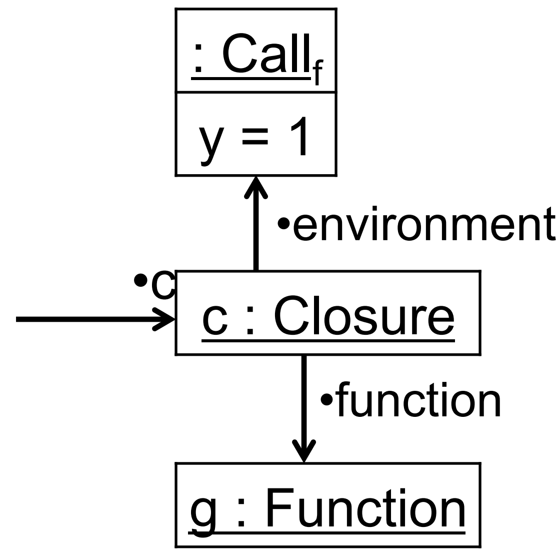
Closure = Function + Environment

- Environment = Old scope chain

```
function f(y) {
  function g() {
    document.write(y);
  }
  return g;
}
var c = f(1);
var d = f(2);
c();
d();
```



Closures




```
function f(y) {
  function g() {
    document.write(y);
  }
  return g;
}
var c = f(1);
var d = f(2);
c(); // prints "1"
d(); // prints "2"
```

- **Closure** `c`

= Function `g(){document.write(y);}`

+ Environment `{y:1}`

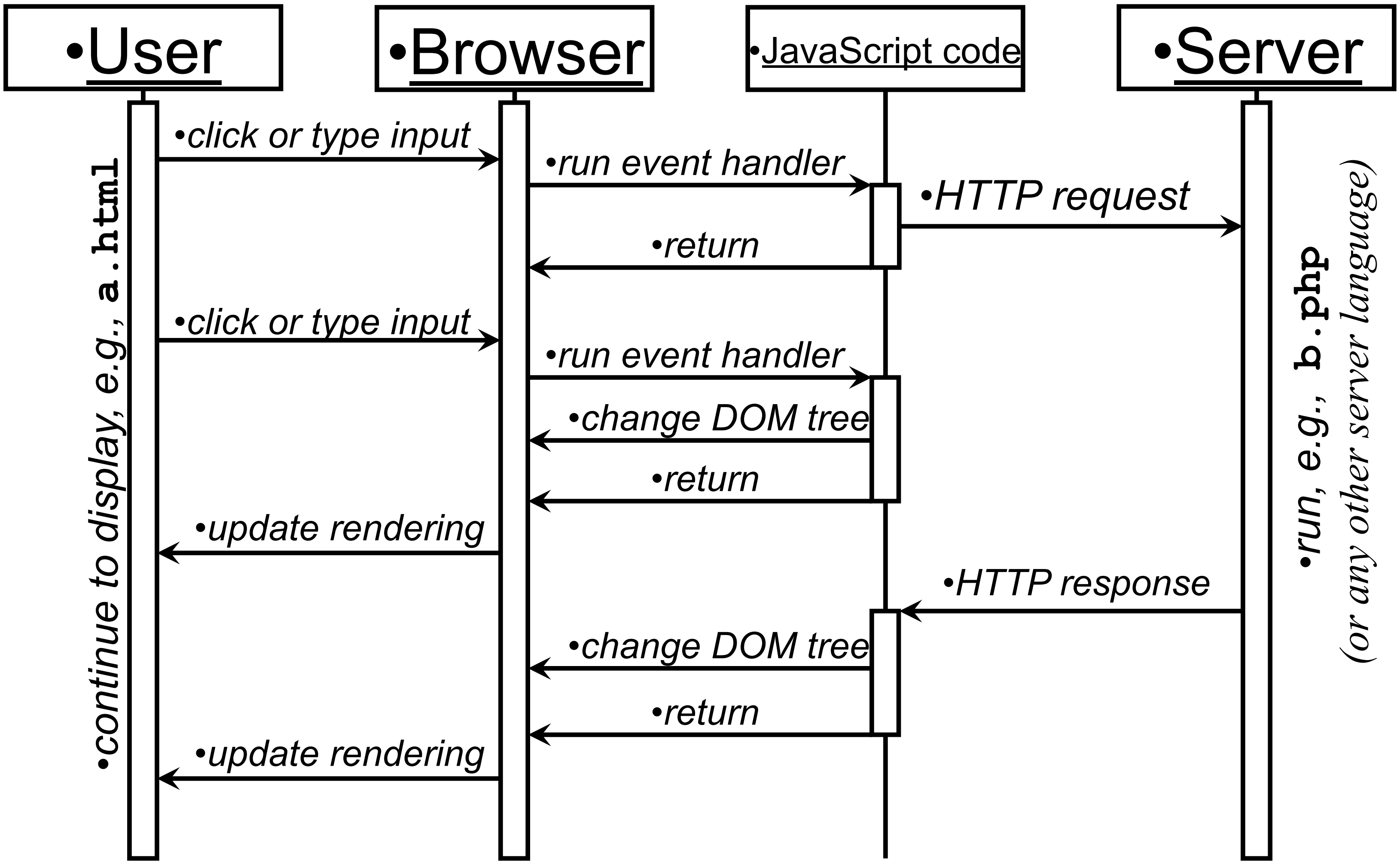
•Soap-box

Why use AJAX?

- AJAX = Asynchronous JavaScript and XML
- Asynchronous = non-blocking =
  send HTTP request to server
  without waiting for the response
- Advantages
  - Large images are not reloaded from server
  - User interface does not freeze up
    ("rich user experience", like non-web-applications)
  - Request-local client-side JavaScript state
    continues to be in scope
- See reading assignment from hw06

•Concepts

AJAX Interaction



XMLHttpRequest Objects

<u>Value properties</u>

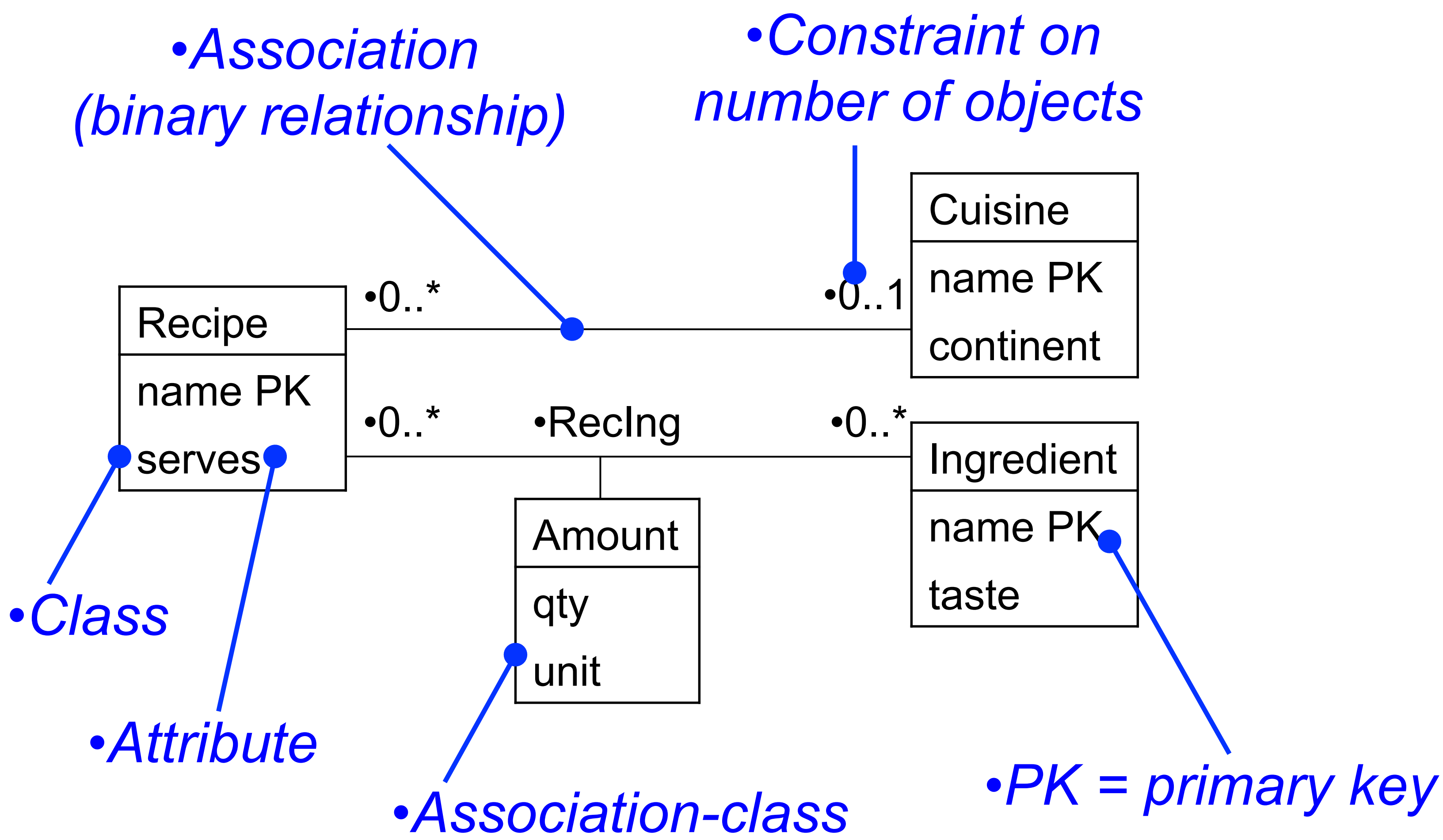
- **`onreadystatechange`**
  `//`*your event handler function*
- **`readyState`**
  `//`0 **`uninitialized`**
  `//`1 **`loading`**
  `//`2 **`loaded`**
  `//`3 **`interactive`**
  `//`4 **`complete`**
- **`responseText`** `//`*string*
- **`responseXML`** `//`*Document*
- **`status`** `//`*int: HTTP code*
- **`statusText`**
  `//`*string: HTTP reason phrase*

<u>Function properties</u>

- **`abort()`**
- **`getAllResponseHeaders()`**
- **`getResponseHeader(parameterName)`**
- **`open(method, url, /*`***bool*****`*/ asynchronous)`**
- **`send(content)`**
- **`setRequestHeader(parameterName, parameterValue)`**

•Concepts

UML with Associations



Administrative

Last Slide

- Good luck!
- Bring a pencil or pen
- Closed book. No notes, phones, etc.