Construction Techniques for Domain-Specific Languages

Brian Guthrie
ThoughtWorks®
bguthrie@thoughtworks.com

properties

http://www.flickr.com/photos/frozenchipmunk/52753779/

“a computer programming language of
limited expressiveness
focused on a particular domain.”

“a computer programming language of
limited expressiveness
focused on a particular domain.”

"a computer programming language of
limited expressiveness
focused on a particular domain."

"iced quad venti (with whip) skinny *caramel* macchiato"

"a computer programming language of
limited expressiveness
focused on a particular domain."

“a computer programming language of
limited expressiveness
focused on a particular domain.”

```
SELECT * FROM pirates
INNER JOIN treasure_chests ON treasure_chests.pirate_id = pirates.id
INNER JOIN islands ON pirates.island_of_residence_id = islands.id
WHERE pirates.name = "Guybrush Threepwood"
AND islands.name = "Melée Island"
```

“a computer programming language of limited expressiveness focused on a particular domain.”

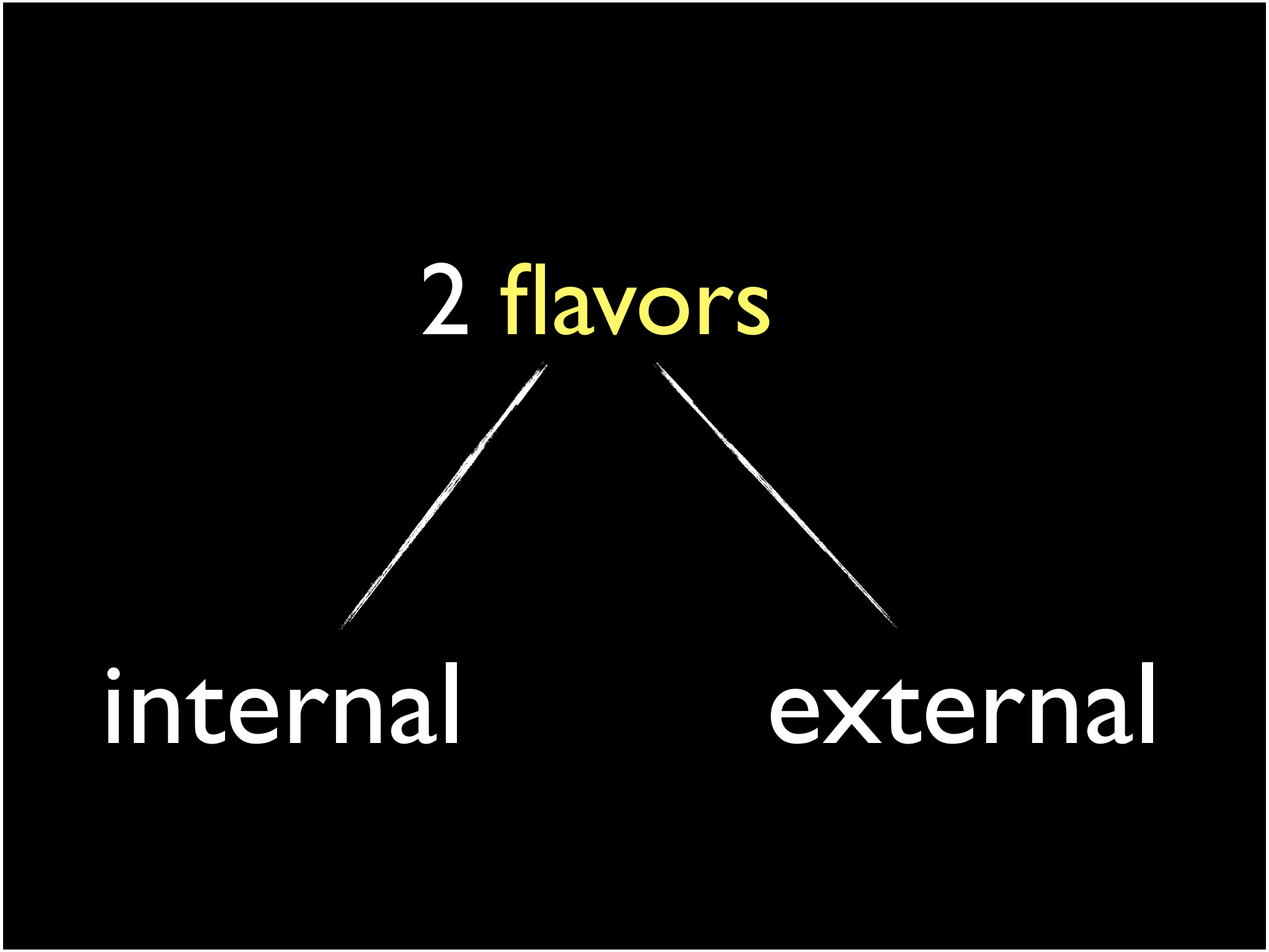
```
<target name="test" depends="compile" description="Runs tests">
  <javac
    destdir="${classes.dir}"
    debug="true"
    source="${javac.version}"
    target="${javac.version}">
    <classpath refid="build.classpath"/>
   <src path="${test.dir}"/>
   <include name="**/*.java"/>
  </javac>
</target>
```

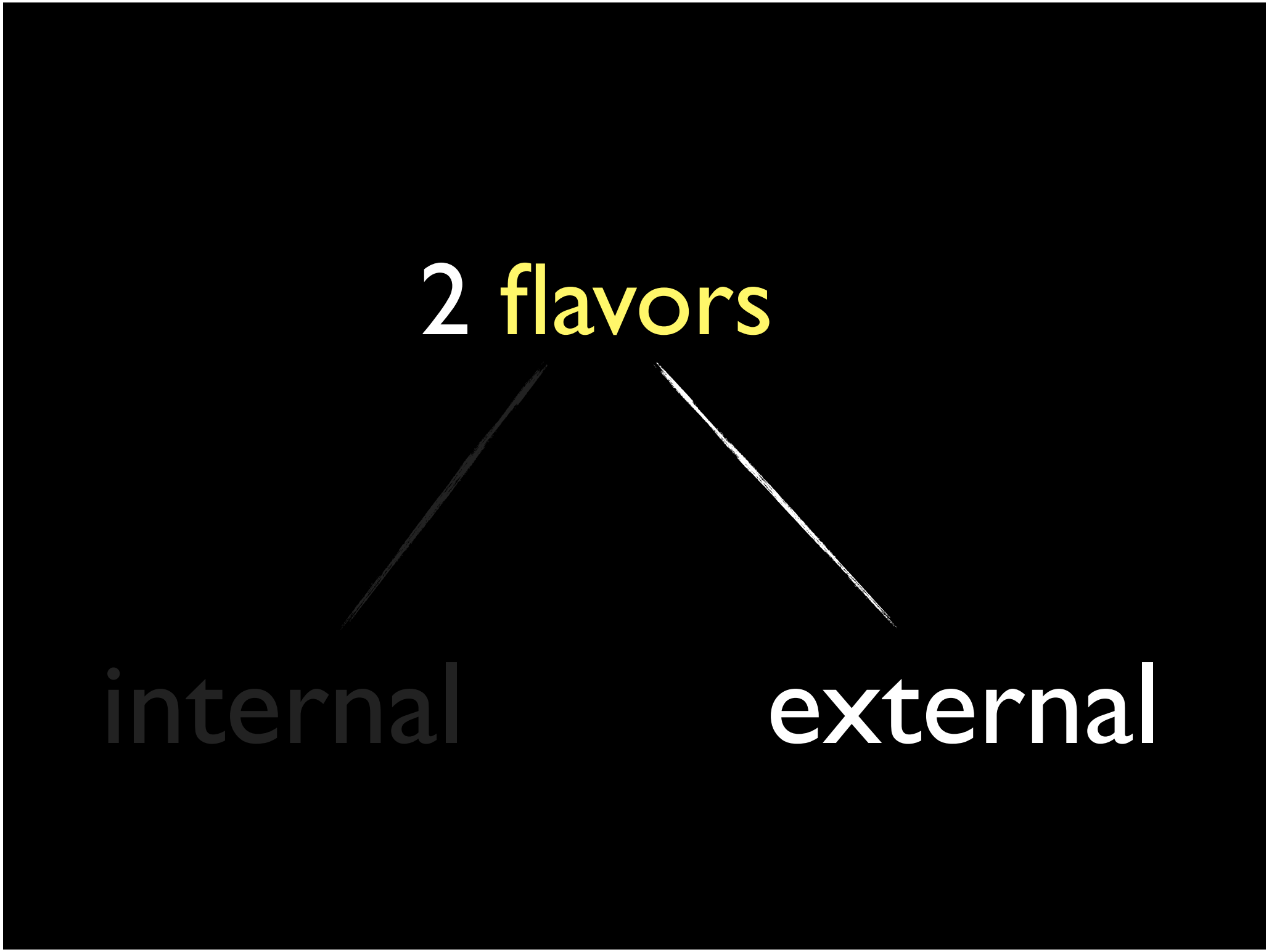
“a computer programming language of limited expressiveness focused on a particular domain.”

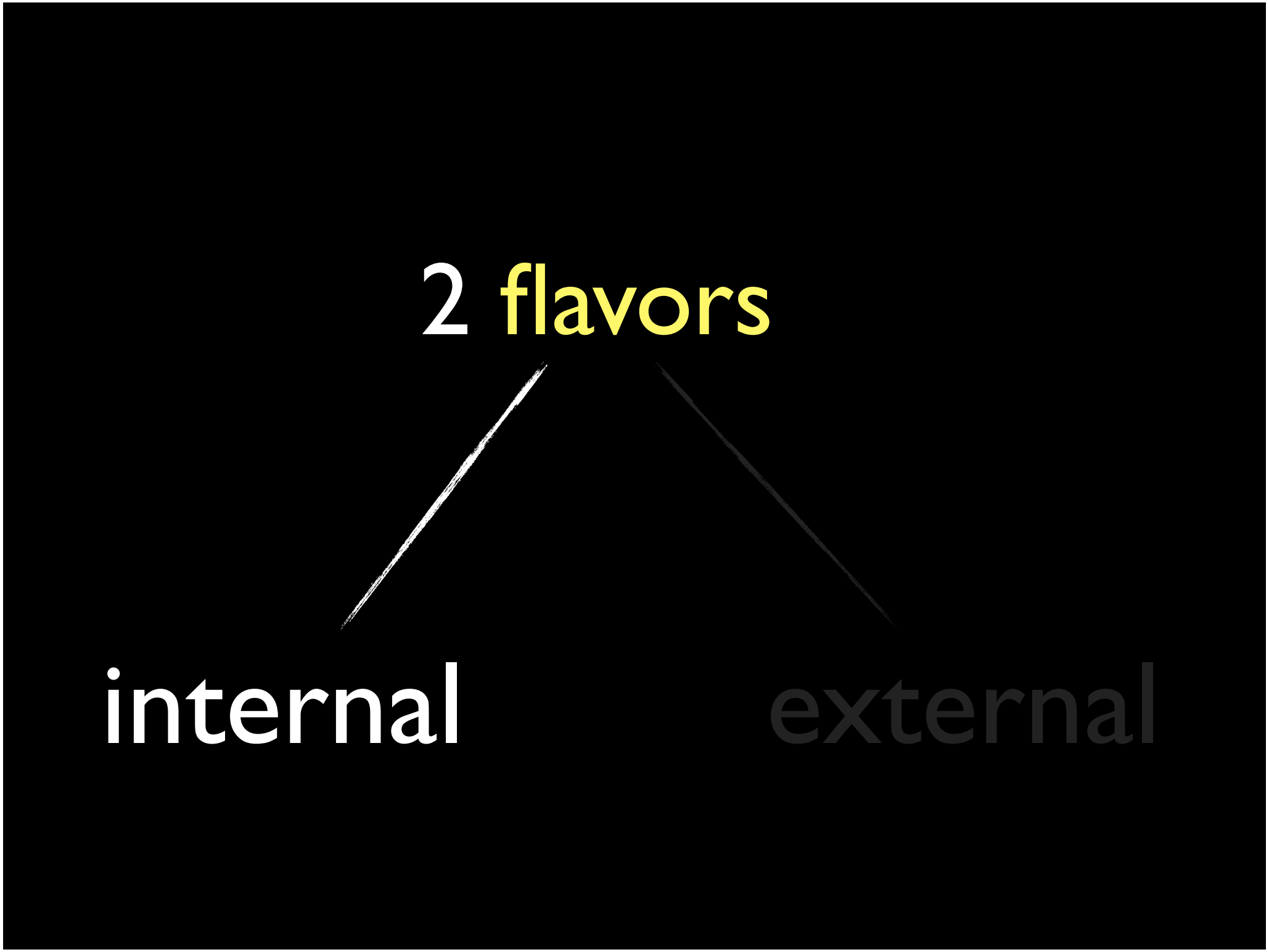
```
<target name="test" depends="compile" description="Runs tests">
  <javac
    destdir="${classes.dir}"
    debug="true"
    source="${javac.version}"
    target="${javac.version}">
    <classpath refid="build.classpath"/>
   <src path="${test.dir}"/>
   <include name="**/*.java"/>
  </javac>
</target>
```

“a computer programming language of limited expressiveness focused on a particular domain.”

```
<target name="test" depends="compile" description="Runs tests">
  <javac
    destdir="${classes.dir}"
    debug="true"
    source="${javac.version}"
    target="${javac.version}">
    <classpath refid="build.classpath"/>
   <src path="${test.dir}"/>
   <include name="**/*.java"/>
  </javac>
</target>
```


2 flavors
internal
external


2 flavors
internal
external


2 flavors
internal
external


API

expressiveness

---

implementation details

fluent interface

a behavior capable of

relaying

or

maintaining

the

instruction context

for a series of method calls

methods make little sense
out of context

[example]

```
Calendar fourPM = Calendar.getInstance();
fourPM.set(Calendar.HOUR_OF_DAY, 16);
Calendar fivePM = Calendar.getInstance();
fivePM.set(Calendar.HOUR_OF_DAY, 17);

AppointmentCalendar calendar = new AppointmentCalendar();

calendar.add("Dentist")
        .from(fourPM)
        .to(fivePM)
        .at("123 N Southwest Ave");

calendar.add("Halloween Party")
        .at(eightPM);

displayAppointments(calendar);
```

a good candidate

```
import fourteen.lines.Elided;

public class ParseAnchorsTest {
    public static void main(String[] args) throws Exception {
        UserAgentContext uacontext = new SimpleUserAgentContext();
        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        DocumentBuilder builder = factory.newDocumentBuilder();
        URL url = new URL("http://www.google.com");
        InputStream in = url.openConnection().getInputStream();
        try {
            Reader reader = new InputStreamReader(in, "ISO-8859-1");
            Document document = builder.newDocument();
            HtmlParser parser = new HtmlParser(uacontext, document);
            parser.parse(reader);
            XPath xpath = XPathFactory.newInstance().newXPath();
            NodeList nodeList = (NodeList) xpath.evaluate("html//a", document, XPathConstants.NODESET);
            int length = nodeList.getLength();
            for(int i = 0; i < length; i++) {
                Element element = (Element) nodeList.item(i);
                System.out.println("## Anchor: " + element.getAttribute("href"));
            }
        } finally {
            in.close();
        }
    }
}
```

source: http://lobobrowser.org/cobra/java-html-parser.jsp

forces

CHOICES

early and often

be

DECLARATIVE

be

accepting

every API is a conversation

http://www.flickr.com/photos/11739182@N03/1263985679/

with *yourself*

with *your team members*

with *your business*


patterns
http://www.flickr.com/photos/mukluk/477913951/

example one

method chaining | type transmogrification

example one

method chaining | type transmogrification

make modifier methods return the host object so that multiple modifiers can be invoked in a single expression.

example one

method chaining | type transmogrification

transform types as needed as part of a fluent interface call

Calendar


the goal
http://www.flickr.com/photos/keylosa/184606430/

```
Calendar fourPM = Calendar.getInstance();
fourPM.set(Calendar.HOUR_OF_DAY, 16);
Calendar fivePM = Calendar.getInstance();
fivePM.set(Calendar.HOUR_OF_DAY, 17);

AppointmentCalendar calendar = new AppointmentCalendar();

calendar.add("Dentist")
        .from(fourPM)
        .to(fivePM)
        .at("123 N Southwest Ave");

calendar.add("Halloween Party")
        .at(eightPM);

displayAppointments(calendar);
```

```
public class AppointmentCalendar {
        private List<Appointment> appointments;

        public AppointmentCalendar() {
                this.appointments = new ArrayList<Appointment>();
        }

        public Appointment add(String name) {
                Appointment appointment = new Appointment(name);
                appointments.add(name);
                return appointment;
        }
}
```

```
public class Appointment {
        private String name;
        private String location;
        private Calendar startTime;
        private Calendar endTime;

        public Appointment at(String location) {
                this.location = location;
                return this;
        }

        public Appointment from(Calendar startTime) {
                this.startTime = startTime;
                return this;
        }

        public Appointment to(Calendar endTime) {
                this.endTime = endTime;
                return this;
        }
}
```

Mocha

what does expects return?

```
car = Car.new
car.expects(:gasoline).
  with("hose", :grade => 83).
  returns(FullTank.new)
car.should be_full
```

```ruby
module Mocha
  class Mock
    def expects(method_name_or_hash, backtrace = nil)
      iterator = ArgumentIterator.new(method_name_or_hash)
      iterator.each { |*args|
        method_name = args.shift
        ensure_method_not_already_defined(method_name)
        expectation = Expectation.new(self, method_name, backtrace)
        expectation.returns(args.shift) if args.length > 0
        @expectations.add(expectation)
      }
    end
  end
end
```

mocha/mock.rb

(`ArgumentIterator#each` returns the Expectation-trust me)

```ruby
module Mocha
  class Expectation
    def returns(*values)
      @return_values += ReturnValues.build(*values)
      self
    end
  end
end
```

mocha/expectation.rb

the finishing problem

where does it all end?
oh, the humanity!

implied contract

type transformations are hidden from the caller
the system will arrive at some desired state

example two

nested closures | semantic model

example two

nested closures | semantic model

express statement sub-
elements of a function call by putting them
into a closure in an argument

example two

nested closures | semantic model

the domain model underpinning
the DSL


Awsymandias
look on my racks, oh ye mighty, and despair.
http://github.com/bguthrie/awsymandias
http://www.flickr.com/photos/stuckincustoms/197905054/

define an environment

persist metadata to S3

spin up in EC2

```
stack.launch unless stack.launched? || stack.running?
```

```ruby
# Give the stack a name, and describe its members.
stack = Awsymandias::EC2::ApplicationStack.define("test") do |s|
  s.instance :db,  :instance_type => "m1.large", ...
  s.instance :app, :instance_type => "c1.xlarge", ...
  s.volume :data, :volume_id => "vol-12345", :instance => :db, ...
end
```

backed by a rich
domain model

```ruby
# Give the stack a name, and describe its members.
stack = Awsymandias::EC2::ApplicationStack.define("test") do |s|
  s.instance :db,  :instance_type => "m1.large", ...
  s.instance :app, :instance_type => "c1.xlarge", ...
  s.volume :data, :volume_id => "vol-12345", :instance => :db, ...
end
```

```ruby
module Awsymandias
  class ApplicationStack
    class << self
      def define(name, &block)
        definition = StackDefinition.new(name)
        yield definition if block_given?
        definition.build_stack
      end
    end
  end
end
```

```ruby
module Awsymandias
  class StackDefinition
    ...

    def instance(name, config={})
      extract_roles(config).each { |r| role(r, name) }
      @defined_instances[name.to_s] = config
    end
  end
end
```

IMPORTANT

definition != domain model


domain model
dsl definition

A DSL is a *language*

parse its inputs!

keep your domain model clean!

```ruby
def launch
  store_app_stack_metadata!

  @unlaunched_instances.each_pair do |instance_name, params|
    @instances[instance_name] = Awsymandias::Instance.launch(params)
    @instances[instance_name].name = instance_name
    @unlaunched_instances.delete instance_name
  end

  ...
end
```

example three

literal extension | string polishing

example three

literal extension | string polishing

add methods to program literals
"monkey patching"

example three

literal extension | string polishing

simple string substitutions to convert
nearly *code to* actual *code*

the goal

```
Page.visit("http://www.digg.com").search(".news-summary").map { elt ->
  NewsStory.new(
    summary: elt.search(".body").first().text(),
    link: elt.search(".offsite").first().attributes.get("href")
  )
}
```

digg

groovy already extends Node

DOMCategory

(brief digression)

```
42.grams.of(Flour)
5.hours.from_now
assert 2.heads > 1.head
```

```ruby
class Integer
  def hours
    self.minutes * 60
  end

  def minutes
    self * 60
  end

  def seconds
    self
  end
end
```

shotgun approach to
open classes

“all willy-nilly”


PLEASE BE SAFE.
Do not stand, sit, climb or
lean on fences.
If you fall, animals could eat you
and that might make them sick.
Thank you.
http://www.flickr.com/photos/tenioman/2771903123/

```
>> 5.method(:hours)
=> #<Method: Fixnum(Integer)#hours>
```

irb

```
>> 5.method(:hours)
=> #<Method: Fixnum(ActiveSupport::CoreExtensions::Numeric::Time)#hours>
```

script/console (in rails)

```ruby
module ActiveSupport #:nodoc:
  module CoreExtensions #:nodoc:
    module Numeric #:nodoc:
      module Time
        def seconds
          ActiveSupport::Duration.new(self, [[:seconds, self]])
        end
        alias :second :seconds

        def minutes
          ActiveSupport::Duration.new(self * 60, [[:seconds, self * 60]])
        end
        alias :minute :minutes

        def hours
          ActiveSupport::Duration.new(self * 3600, [[:seconds, self * 3600]])
        end
        alias :hour :hours
      end
    end
  end
end
```

back to Groovy...

```
use (DOMCategory) {
    Element
    NodeList
}
```

children(), attributes(), text(), name(), parent(), depthFirst(), breadthFirst()

size(), list(), text(), child

this is awesome!

and speaking of which...

```
calendar.add("Dentist")
   .from(4.pm)
   .to(5.pm)
   .at("123 N Southwest Ave");

calendar.add("Halloween Party")
   .at(8.pm);
```

we can do better!

```
class IntegerWithTimeSupport {
  static Integer getAm(Integer self) {
    self == 12 ? 0 : self
  }

  static Integer getPm(Integer self) {
    self == 12 ? 12 : self + 12
  }
}
```

```
use(IntegerWithTimeSupport) {
  calendar.add("Dentist")
    .from(4.pm)
    .to(5.pm)
    .at("123 N Southwest Ave")

  calendar.add("Halloween Party")
    .at(8.pm)
}
```

```
class Page {
  def static visit(url) {
    // Pseudo-code alert!
    return new HTMLParser(
      new URL(url).inputStream
    ).parse()
  }
}
```

search?

```
class DOMSearchCategory {
  def static search(Node node, cssSelector) {
    XPathFactory.newInstance().newXPath().evaluate(
      new XPathConverter(cssSelector).toXPath(),
      node, XPathConstants.NODESET)
  }
}
```

```
class XPathConverter {
  def static toXPath(String cssSelector) {
    if (cssSelector.startsWith("#")) {
      def rawId = cssSelector.replace('#', '')
      return "//@id = ${rawId}"
    } else if (cssSelector.startsWith(".")) {
      def rawClass = cssSelector.replace('.', '')
      return "//contains(concat(' ', @class, ' '), ' ${rawClass} ')"
    } else { ... }
  }
}
```

```
Page.visit("http://www.digg.com").search(".news-summary").map { elt ->
  NewsStory.new(
    summary: elt.search(".body").first().text(),
    link: elt.search(".offsite").first().attributes.get("href")
  )
}
```

example four

literal extension | method chaining

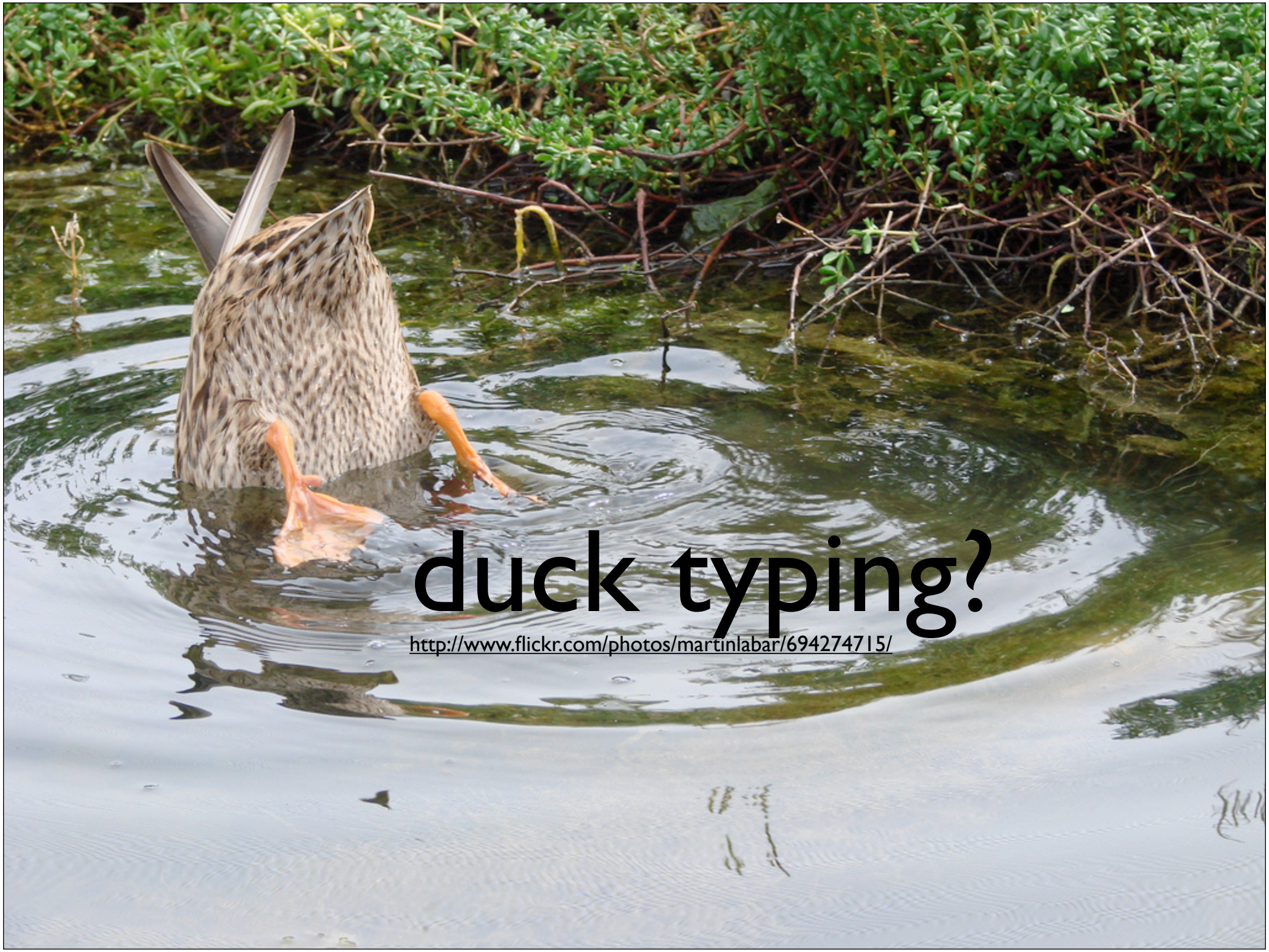
```
$(document).ready(function() {
  $("#navbar .menu").hide().click(function() {
    $(this).show().removeClass("raised").addClass("recessed").css({ height: 300 });
  })
});
```

```
jQuery.fn = jQuery.prototype = {
  init: function( selector, context ) {
    // Make sure that a selection was provided
    selector = selector || document;

    ...

    return this.setArray(jQuery.isArray( selector ) ?
      selector :
      jQuery.makeArray(selector));
  }
}


// Force the current matched set of elements to become
// the specified array of elements
setArray: function( elems ) {
  this.length = 0;
  Array.prototype.push.apply( this, elems );

  return this;
},
```

literal extension at the
object level

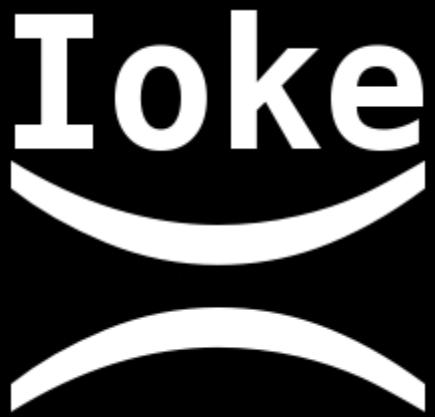
jQuery objects
act like arrays

jQuery objects
are arrays


duck typing?
http://www.flickr.com/photos/martinlabar/694274715/

example five

method chaining andChaining andChaining


Ioke

message chains

```
SomeBaseObject mimic someMethod(arg: anArgument) someOtherMethod
```

mimic: cloning is expected and encouraged

```
Account = Origin mimic do(
  balance = 0.0
  deposit = method(v, self balance += v)
  show = method("Account balance: $#{balance}" println)
)

"Inital: " print
Account show

"Depositing $10" println
Account deposit(10.0)

"Final: " print
Account show
```

“iced quad venti (with whip) skinny *caramel* macchiato”

Macchiato iced quad venti withWhip skinny caramel

```
Espresso = Origin mimic do(
  shots    = 1
  milk     = nil
  syrup    = nil
  iced     = false
  toppings = []

  single = method(mimic)
  double = method(with(shots: self shots + 1))
  triple = method(with(shots: self shots + 2))
  quad   = method(with(shots: self shots + 3))
)
```

```
EspressoDrink = Espresso mimic do(
  short  = tall  = cell(:single)
  grande = venti = cell(:double)

  iced = method(
    self with(shots: self shots + 1)
  )

  sweetened = method(
    self with(syrup: Syrup plain)
  )
)
```

```
Latte = EspressoDrink mimic do(
  milk = Milk steamed

  skinny = method(
    self with(
      milk: Milk skim,
      syrup: Syrup fatFree)
  )

  withWhip = method(
    self with(toppings:
      self toppings + [ Milk whipped ])
  )
)
```


calories
http://www.flickr.com/photos/lifeontheedge/2077384723/
“the leaning tower of bacon”

```
it("should calculate calories",
  Latte grande nonfat calories should == 100
)
```

```
Milk steamed calories = 30
Syrup calories = 80

Espresso calories = method(
  (self shots * 5) + self syrup calories + self milk calories
)
```


http://www.flickr.com/photos/wili/214317898/
bringing
it all
back
home

a DSL is:

- an API
- a language
- a conversation

be

DECLARATIVE

know your audience

fluent interface

MESSAGE CHAINS

nested

`function|closure`

include • use • extend • eval

Brian Guthrie
ThoughtWorks®
bguthrie@thoughtworks.com

questions?