• a sip of •

CoffeeScript

Variables and Functions

• *level 1* •

A Beautiful Programming Language

- Least amount of code to solve problems
- Readable and Understandable
- Easy to Maintain

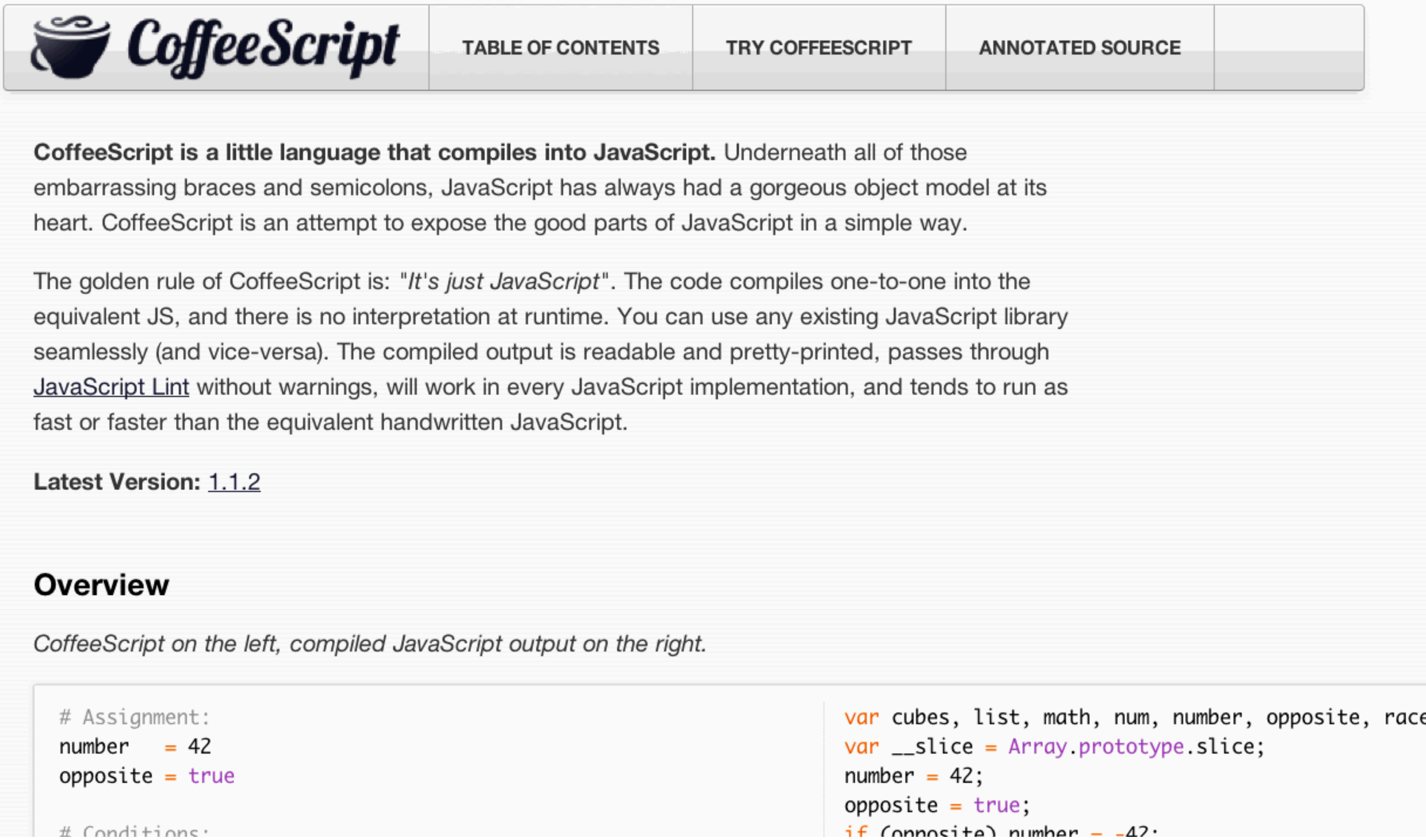
JavaScript ?

CoffeeScript

compiles into *JavaScript*

- Least amount of code to solve problems
- Readable and Understandable
- Easy to Maintain

http://jashkenas.github.com/coffee-script/



CoffeeScript is a little language that compiles into JavaScript. Underneath all of those embarrassing braces and semicolons, JavaScript has always had a gorgeous object model at its heart. CoffeeScript is an attempt to expose the good parts of JavaScript in a simple way.

The golden rule of CoffeeScript is: "*It's just JavaScript*". The code compiles one-to-one into the equivalent JS, and there is no interpretation at runtime. You can use any existing JavaScript library seamlessly (and vice-versa). The compiled output is readable and pretty-printed, passes through JavaScript Lint without warnings, will work in every JavaScript implementation, and tends to run as fast or faster than the equivalent handwritten JavaScript.

Latest Version: 1.1.2

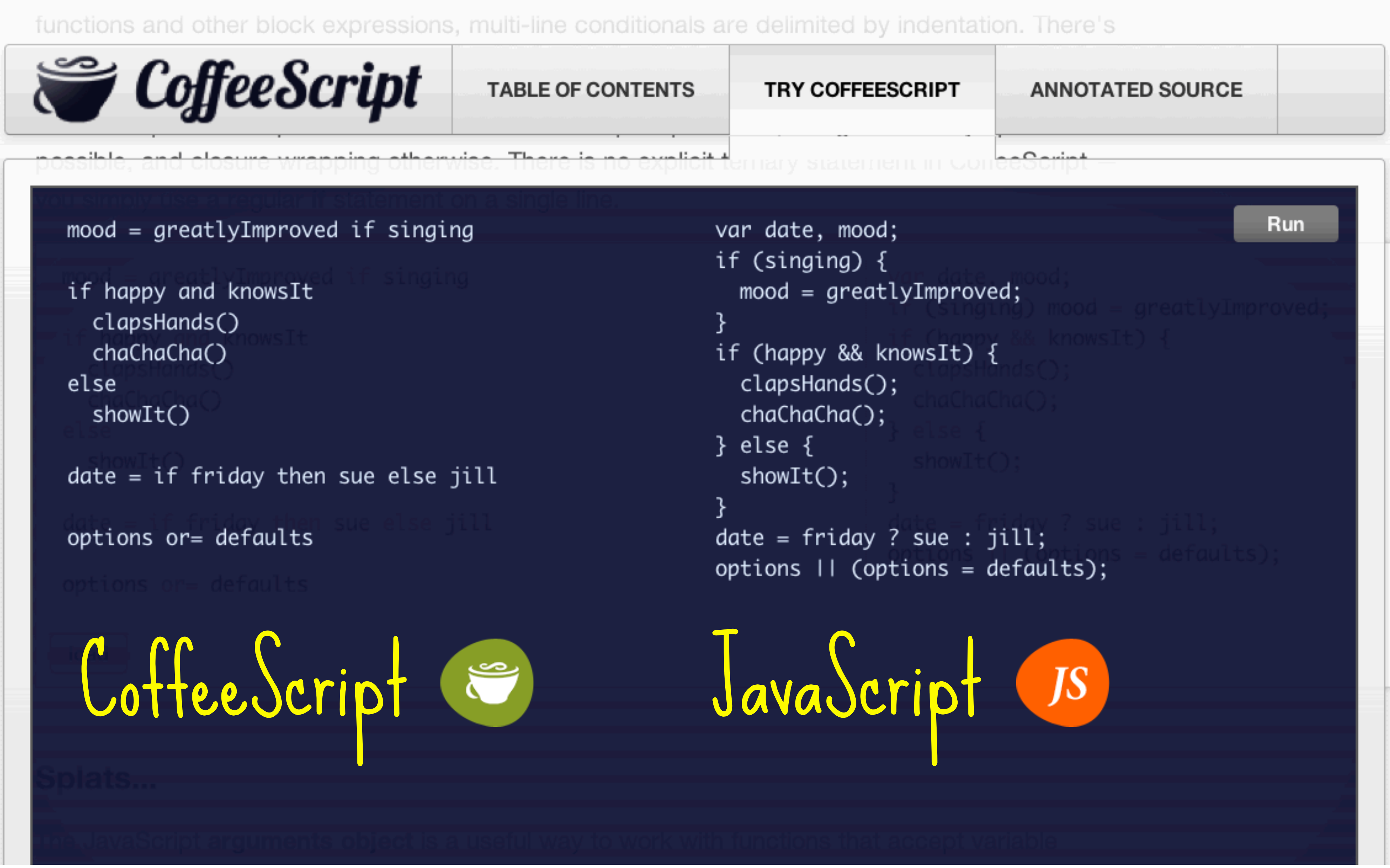
Overview

CoffeeScript on the left, compiled JavaScript output on the right.

```
# Assignment:
number   = 42
opposite = true
```

```
var cubes, list, math, num, number, opposite, race
var __slice = Array.prototype.slice;
number = 42;
opposite = true;
```

http://jashkenas.github.com/coffee-script/



```
mood = greatlyImproved if singing

if happy and knowsIt
  clapsHands()
  chaChaCha()
else
  showIt()

date = if friday then sue else jill

options or= defaults
```

```
var date, mood;
if (singing) {
  mood = greatlyImproved;
}
if (happy && knowsIt) {
  clapsHands();
  chaChaCha();
} else {
  showIt();
}
date = friday ? sue : jill;
options || (options = defaults);
```

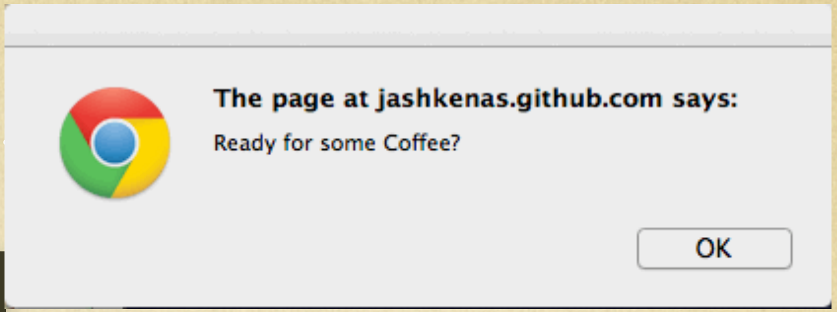
CoffeeScript

JavaScript

CoffeeScript

JavaScript

Variables


The page at jashkenas.github.com says:
Ready for some Coffee?

```coffeescript
message = "Ready for some Coffee?"
alert(message)
```

```javascript
var message;
message = "Ready for some Coffee?";
alert(message);
```

No variable declarations

No semicolons

Variables and Functions

2 ways to create Functions in JS

✦ Named Functions

```
function coffee() {
  return confirm("Ready for some Coffee?");
}
```

JS

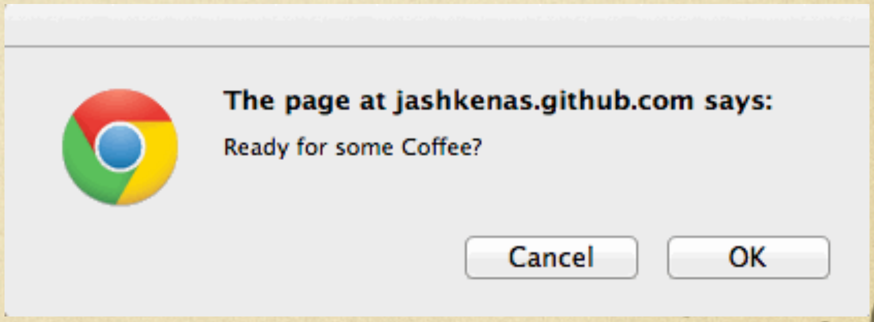
✦ Function Expressions

```
var coffee = function() {
  return confirm("Ready for some Coffee?");
}
```

JS

Both called with

```
coffee();
```



The page at jashkenas.github.com says:

Ready for some Coffee?

Variables and Functions

We only use Function Expressions

```
coffee = ->
  confirm "Ready for some Coffee?"
```

1 tab or 2 spaces indented

-> converts to function() {

Always has a return value

```
var coffee = function() {
  return confirm("Ready for some Coffee?");
}
```

JS

Returning a String

```
coffee = ->
  answer = confirm "Ready for some Coffee?"
  "Your answer is " + answer
```

same as

```
"Your answer is #{answer}"
```

```
var coffee;
coffee = function() {
  var answer;
  answer = confirm("Ready for some Coffee?");
  return "Your answer is " + answer;
}
```

JS

Variables and Functions

Function Parameters

```coffeescript
coffee = (message) ->
  answer = confirm message
  "Your answer is #{answer}"
```

```javascript
var coffee;
coffee = function(message) {
  var answer;
  answer = confirm(message);
  return "Your answer is " + answer;
}
```

Variables and Functions

Calling Functions

```
coffee = ->
```

```
coffee()
```

```
coffee = (message) ->
```

```
coffee("Yo")
```

```
coffee "Yo"
```

parenthesis optional

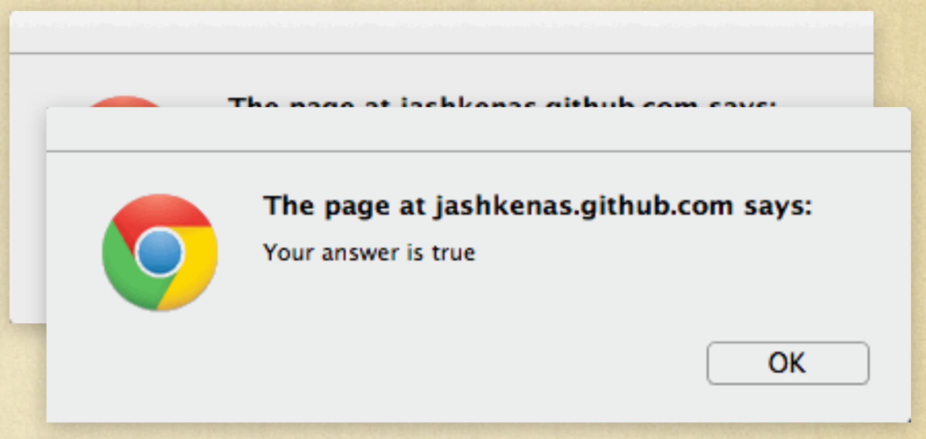
```
coffee = (message, other) ->
```

```
coffee("Yo", 2)
```

```
coffee "Yo", 2
```

Function Parameters

```
coffee = (message) ->
  answer = confirm message
  "Your answer is #{answer}"
```



The page at jashkenas.github.com says:

The page at jashkenas.github.com says:

Your answer is true

```
alert coffee("Ready for some Coffee?")
```

parenthesis on everything

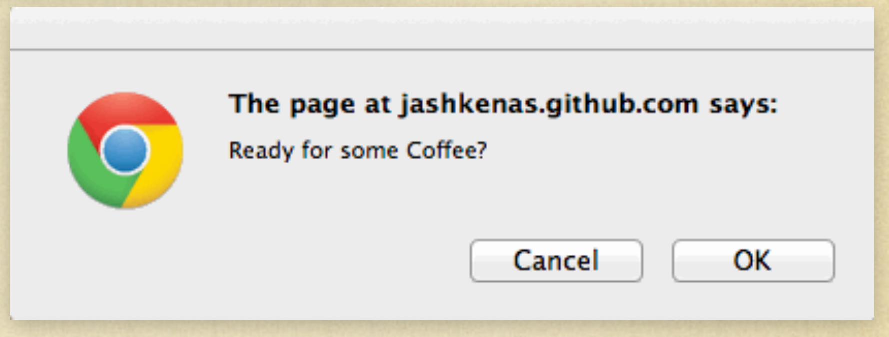
but the outermost call

Optional Parameters

If we want a default message

```
coffee = (message = "Ready for some Coffee?") ->
  answer = confirm message
  "Your answer is #{answer}"
```

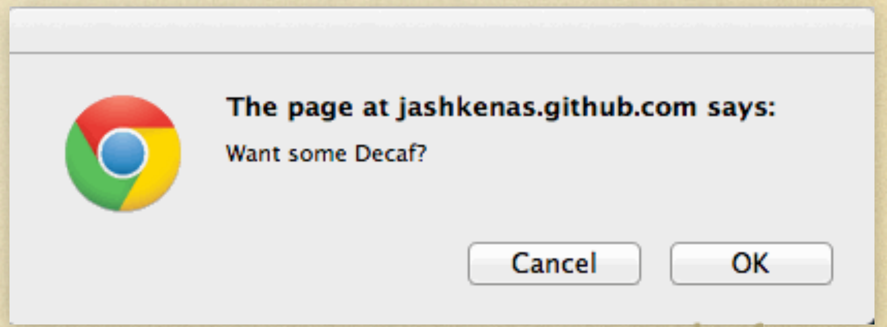
```
alert coffee()
```



The page at jashkenas.github.com says:

Ready for some Coffee?

```
alert coffee("Want some Decaf?")
```



The page at jashkenas.github.com says:

Want some Decaf?

Optional Parameters

```coffeescript
coffee = (message ="Ready for some Coffee?") ->
  answer = confirm message
  "Your answer is #{answer}"
```

```javascript
var coffee;
coffee = function(message) {
  var answer;
  if (message == null) {
    message = "Ready for some Coffee?";
  }
  answer = confirm(message);
  return "Your answer is " + answer;
}
```

Variables and Functions

The CoffeeScript Compiler (optional)

1. Install Node.js http://nodejs.org/
2. Install npm http://npmjs.org/
3.

```
$ npm install -g coffee-script
```

```
$ coffee -h
```

```
Usage: coffee [options] path/to/script.coffee

  -c, --compile      compile to JavaScript and save as .js files
  -i, --interactive  run an interactive CoffeeScript REPL
  -o, --output       set the directory for compiled JavaScript
  -w, --watch        watch scripts for changes, and recompile
  -p, --print        print the compiled JavaScript to stdout
  -e, --eval         compile a string from the command line
```

Command Line Examples

```
$ coffee -c test.coffee
```

*Creates **test.js***

```
$ coffee -cw test.coffee
```

Every time test.coffee is updated re-compile.

```
$ coffee -c src -o js
```

*Compile all .coffee files in the **src dir** into the **js dir**.*

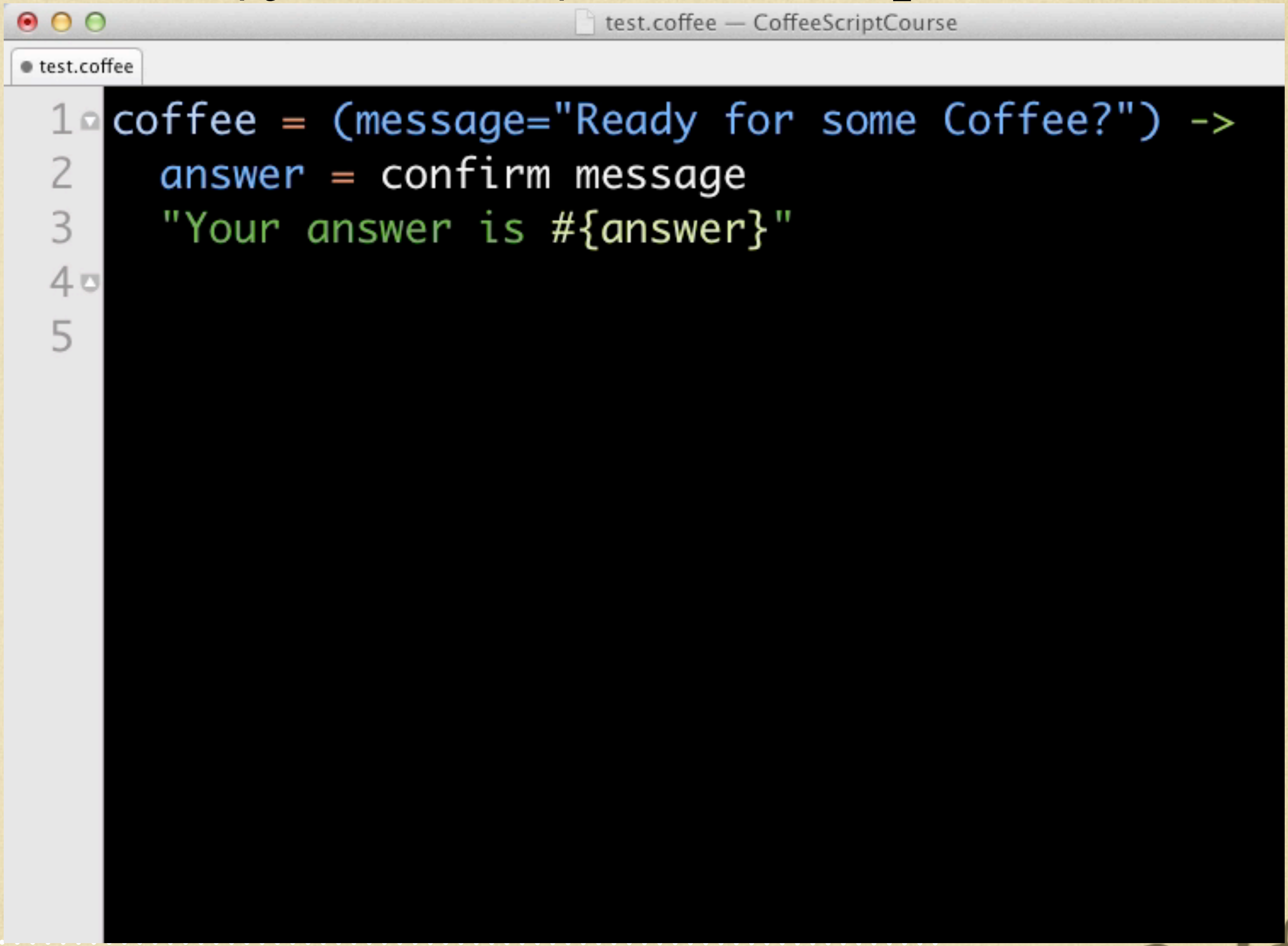
```
$ coffee -wc src -o js
```

Every time a file is updated re-compile.

CoffeeScript TextMate Bundle

https://github.com/jashkenas/coffee-script-tmbundle

& Sublime Text 2



```
coffee = (message="Ready for some Coffee?") ->
  answer = confirm message
  "Your answer is #{answer}"
```

Variables and Functions

CoffeeScript

Creative Commons

name	author	URL
Information density	Arenamontanus	http://www.flickr.com/photos/arenamontanus/535807315
Paper Writings: Quotes	Lucia..	http://www.flickr.com/photos/angelic0devil6/2104607220
Sütterlin	ninastoessinger	http://www.flickr.com/photos/ninastoessinger/4713004390
FF DIN Round: ‘Round Pie	FontFont	http://www.flickr.com/photos/fontfont/4840584146
Sonnet 18	Jinx!	http://www.flickr.com/photos/span112/3041263205

• a sip of •

CoffeeScript

download the slides

use the hints

Applied jQuery

• *level 2* •

jQuery to CoffeeScript

```
jQuery(function($) {

  function changeTab(e) {
    e.preventDefault();
    $("#tabs li a.active").removeClass("active");
    $(this).addClass("active");
  }

  $("#tabs ul li a").click(changeTab);
});
```

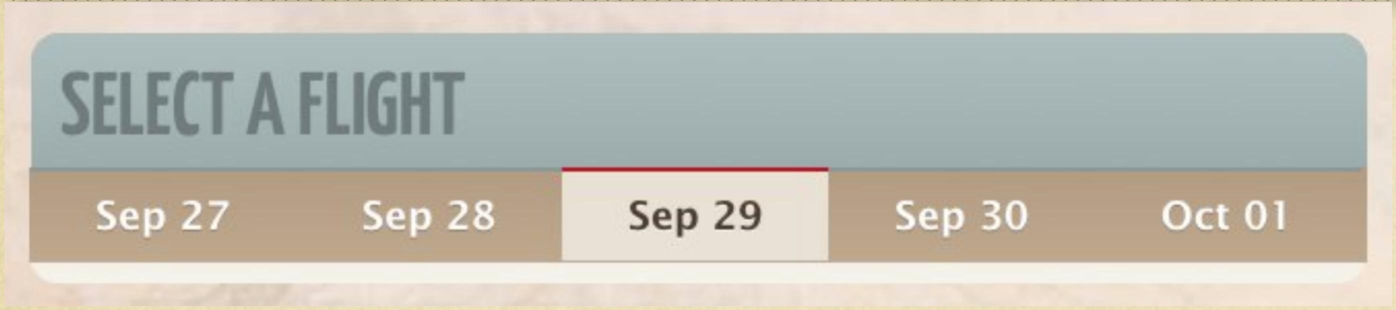
```
jQuery ($) ->
```

or

```
$ ->
```

*

JS



* If no other libraries are using $

jQuery to CoffeeScript

```
$ ->

  function changeTab(e) {
    e.preventDefault();
    $("#tabs li a.active").removeClass("active");
    $(this).addClass("active");
  }

  $("#tabs ul li a").click(changeTab);
});
```

```
changeTab = (e) ->
```

JS

Applied jQuery

jQuery to CoffeeScript

```
$ ->
  changeTab = (e) ->

    e.preventDefault();
    $("#tabs li a.active").removeClass("active");
    $(this).addClass("active");
  }

  $("#tabs ul li a").click(changeTab);
});
```

Remove all semicolons and curly brackets

Applied jQuery

jQuery to CoffeeScript

```
$ ->
  changeTab = (e) ->
    e.preventDefault()
    $("#tabs li a.active").removeClass("active")
    $(this).addClass("active")

  $("#tabs ul li a").click(changeTab)
```

Optionally remove parenthesis

jQuery to CoffeeScript

```
$ ->
  changeTab = (e) ->
    e.preventDefault()
    $("#tabs li a.active").removeClass "active"
    $(this).addClass "active"

  $("#tabs ul li a").click changeTab
```

same as

```
$(@).addClass "active"
```

@ = this

jQuery to CoffeeScript

```
jQuery(function($) {
  function changeTab(e) {
    e.preventDefault();
    $("#tabs li a.active").removeClass("active");
    $(this).addClass("active");
  }
  $("#tabs ul li a").click(changeTab);
});
```

```
$ ->
  changeTab = (e) ->
    e.preventDefault()
    $("#tabs li a.active").removeClass "active"
    $(@).addClass "active"

  $("#tabs ul li a").click changeTab
```

jQuery to CoffeeScript

```
$("#tabs #error a").click(function (e){
  e.preventDefault();
});
```

JS

```
$("#tabs #error a").click (e) ->
  e.preventDefault()
```

```
$('#confirm').queue(function() {
  $(this).dequeue();
});
```

JS

```
$("#confirm").queue ->
  $(@).dequeue()
```

Applied jQuery

jQuery to CoffeeScript

```
function showNumberOfFlights(e) {
  var num_flights = $(this).data('flights');
  $(this).append("<span>" + num_flights  +"</span>");
  $("#tabs span.tooltip").show();
}
```

```
showNumberOfFlights = (e) ->
  num_flights = $(@).data 'flights'
  $(@).append "<span>#{flights}</span>"
  $("#tabs span.tooltip").show()
```

• a sip of •

CoffeeScript

Conditionals & Operators

• *level 3* •

If Statement

```
if (age < 18) {
  alert('Under age');
}
```

JS

```
if age < 18
  alert 'Under age'
```

```
alert 'Under age' if age < 18
```

```
if age < 18 then alert 'Under age'
```

Parenthesis
Optional

Conditionals & Operators

If Else Statement

```
if (age < 18) {
  alert('Under age');
} else {
  alert('of age');
}
```

JS

```
if age < 18
  alert 'Under age'
else
  alert 'of age'
```

```
if age < 18 then alert 'Under age' else alert 'of age'
```

No Ternary Operator

Conditionals & Operators

Operators

CoffeeScript	JavaScript
`==` `is`	`===`
`!=` `isnt`	`!==`
`not`	`!`
`and`	`&&`
`or`	`||`
`true` `yes` `on`	`true`
`false` `no` `off`	`false`

Operator Examples & Unless

```
if paid() and coffee() is on then pour()
```

```
if (paid() && coffee() === true) {
  pour();
}
```

```
addCaffeine() if not Decaf()
```

```
addCaffeine() unless Decaf()
```

Chained Comparisons

```javascript
if (2 < newLevel && newLevel < 5) {
  alert("In Range!");
}
```

```coffeescript
if 2 < newLevel < 5
 alert "In Range!"
```

Switch Statements

JS

```
var message = (function() {
  switch (cupsOfCoffee) {
    case 0:
      return 'Asleep';
    case 1:
      return 'Eyes Open';
    case 2:
      return 'Buzzed';
    default:
      return 'Dangerous';
  }
})();
```

```
message = switch cupsOfCoffee
  when 0 then 'Asleep'
  when 1 then 'Eyes Open'
  when 2 then 'Buzzed'
  else 'Dangerous
```

Existential Operators

How do we check to see that `cupsOfCoffee` isn't defined and isn't null?

JS

```
if (typeof cupsOfCoffee !== "undefined" && cupsOfCoffee !== null) {
  alert('it exists!');
}
```

```
if cupsOfCoffee?
  alert 'it exists!'
```

```
alert 'it exists!' if cupsOfCoffee?
```

Conditionals & Operators

Existential Operators

Set `cupsOfCoffee` to Zero unless previously set

```
if not cupsOfCoffee?
  cupsOfCoffee = 0
```

```
cupsOfCoffee = 0 unless cupsOfCoffee?
```

```
cupsOfCoffee ?= 0
```

Existential Operators

Call `brew()` on `coffeePot` only if it exists

```
if coffeePot?
  coffeePot.brew()
```

```
coffeePot?.brew()
```

Only call function if it exists

```
vehicle.start_engine?().shift_gear?()
```

in Ruby "try()"

• a sip of •

CoffeeScript

Arrays, Objects, Iteration

• *level 4* •

Ranges

```
range = [1..4]
```

```
var range = [1, 2, 3, 4];
```

```
range = [1...4]
```

```
var range = [1, 2, 3];
```

With three dots excludes the end

Variables & Subsets

```
start = 5
end = 10

range = [start..end]
```

```
[5, 6, 7, 8, 9, 10]
```

```
range[1..4]
```

```
[6, 7, 8, 9]
```

```
range[1...range.length]
```

```
[6, 7, 8, 9, 10]
```

```
range[1..-1]
```

Arrays, Objects, Iteration

Arrays

```
storeLocations = ['Orlando', 'Winter Park', 'Sanford']
```

Can use new lines instead of commas

```
storeLocations = [
  'Orlando'
  'Winter Park'
  'Sanford'
]
```

Loops

```
storeLocations = ['Orlando', 'Winter Park', 'Sanford']
```

```
storeLocations.forEach (location, index) ->
  alert "Location: #{location}"
```

```
storeLocations.forEach(function(location, index) {
  return alert("Location: " + location);
});
```

Loops

```
storeLocations = ['Orlando', 'Winter Park', 'Sanford']
```

```
storeLocations.forEach (location, index) ->
  alert "Location: #{location}"
```

```
for location in storeLocations
  alert "Location: #{location}"
```

```
alert "Location: #{location}" for location in storeLocations
```

This is a list comprehension

List Comprehensions

it's an expression

```
storeLocations = ['Orlando', 'Winter Park', 'Sanford']
```

Add ", FL" to each storeLocation

```
"#{loc}, FL" for loc in storeLocations
```

```
['Orlando, FL', 'Winter Park, FL', 'Sanford, FL']
```

```
storeLocations = ("#{loc}, FL" for loc in storeLocations)
```

the parenthesis are important

```
geoLocate(loc) for loc in storeLocations when loc isnt 'Sanford'
```

filter (expression)

List Comprehensions

it's an expression

```
storeLocations = ['Orlando', 'Winter Park', 'Sanford']
```

Create new array without Sanford

```
['Orlando', 'Winter Park']
```

```
newLocs = []
for loc in storeLocations
  newLocs.push loc if loc isnt 'Sanford'
```

same as

```
newLocs = (loc for loc in storeLocations when loc isnt 'Sanford')
```

Splats For a variable number of arguments

```
searchLocations = (brand, cities...) ->
  "looking for #{brand} in #{cities.join(',')}"
```

```
searchLocations 'Starducks', 'Orlando'
```

```
'Looking for Starducks in Orlando'
```

```
searchLocations 'Starducks', 'Orlando', 'Winter Park'
```

```
'Looking for Starducks in Orlando, Winter Park'
```

same as

```
params = ['Starducks', 'Orlando', 'Winter Park']
searchLocations(params...)
```

Objects

Objects are lists of keys & values (hash)

```
coffee = { name: 'French', strength: 1 }
```

curly braces optional

```
coffee = name: 'French', strength: 1
```

commas optional

```
coffee =
  name: 'French'
  strength: 1
```

Objects

```coffeescript
coffee =
  name: 'French'
  strength: 1
  brew: -> alert("brewing #{@name}")
```

called with

```coffeescript
coffee.brew()
```

```javascript
var coffee = {
  name: 'French',
  strength: 1,
  brew: function() {
    return alert("brewing " + this.name);
  }
};
```

Objects

```coffeescript
coffee =
  name: 'French'
  strength: 1
  brew: -> alert("brewing #{@name}")
  pour: (amount=1) ->
    if amount is 1
      "Poured a single cup"
    else
      "Poured #{amount} cups"
```

```javascript
pour: function(amount) {
  if (amount == null) amount = 1;
  if (amount === 1) {
    return "Poured a single cup";
  } else {
    return "Poured " + amount + " cups";
  }
```

JS

Careful with your Indenting!

```
coffee =
  name: 'French'
strength: 1
```

indent issues!

```
coffee = {
  name: 'French'
};

({
  strength: 1
})
```

Complex Objects

```
coffees =
  french:
    strength: 1
    in_stock: 20
  italian:
    strength: 2
    in_stock: 12
  decaf:
    strength: 0
    in_stock: 8
```

```
var coffees = {
  french: {
    strength: 1,
    in_stock: 20
  },
  italian: {
    strength: 2,
    in_stock: 12
  },
  decaf: {
    strength: 0,
    in_stock: 0
  }
};
```

```
coffees.italian.in_stock
```

```
12
```

Object Iteration with of

KEY VALUE

```
"#{coffee} has #{attrs.in_stock}" for coffee, attrs of coffees
```

iterating over object

```
["french has 20", "italian has 12", "decaf has 0"]
```

```
coffees =
  french:
    strength: 1
    in_stock: 20
  italian:
    strength: 2
    in_stock: 12
  decaf:
    strength: 0
    in_stock: 0
```

Object Iteration with of

```
"#{coffee} has #{attrs.in_stock}" for coffee, attrs of coffees
```

same as `["french has 20", "italian has 12", "decaf has 0"]`

```
for coffee, attrs of coffees
  "#{coffee} has #{attrs.in_stock}"
```

```
to_print = for coffee, attrs of coffees when attrs.in_stock > 0
  "#{coffee} has #{attrs.in_stock}"
to_print.join ", "
```

```
"french has 20, italian has 12"
```

Object Iteration with of

```
to_print = for coffee, attrs of coffees when attrs.in_stock > 0
  "#{coffee} has #{attrs.in_stock}"
to_print.join ", "
```

```
var attrs, coffee, to_print;

to_print = (function() {
  var _results;
  _results = [];
  for (coffee in coffees) {
    attrs = coffees[coffee];
    if (attrs.in_stock > 0) _results.push("" + coffee + " has "
+ attrs.in_stock);
  }
  return _results;
})();

to_print.join(", ")
```

• a sip of •

CoffeeScript

Applied jQuery, Part 2

• *level 5* •

Object Simplicity

```
$("#tabs ul li a").bind({
  click: changeTab,
  mouseenter: showNumberOfFlights,
  mouseleave: hideNumberOfFlights
});
```

```
$("#tabs ul li a").bind
  click: changeTab
  mouseenter: showNumberOfFlights
  mouseleave: hideNumberOfFlights
```

A Complex Example

```javascript
function showFlights(activeDiv) {
  $("#tabs div").hide();

  if (fetchingFlights) {
    fetchingFlights.abort();
  }

  fetchingFlights = $.ajax('/flights', {
    data: { date: activeDiv },
    cache: false,
    error: function(result) {
      if (result.statusText != "abort") {
        $('#tabs #error').show();
      }
    }
  });
}
```

```coffeescript
showFlights = (activeDiv) ->
  $("#tabs div").hide()
```

JS

Applied jQuery, Part 2

A Complex Example

```
showFlights = (activeDiv) ->
  $("#tabs div").hide()
```

```
  if (fetchingFlights) {
    fetchingFlights.abort();
  }

  fetchingFlights = $.ajax('/flights', {
    data: { date: activeDiv },
    cache: false,
    error: function(result) {
      if (result.statusText != "abort") {
        $('#tabs #error').show();
      }
    }
  });
}
```

```
if fetchingFlights
  fetchingFlights.abort()
```

JS

Applied jQuery, Part 2

a sip of CoffeeScript

A Complex Example

```
showFlights = (activeDiv) ->
  $("#tabs div").hide()

  if fetchingFlights
    fetchingFlights.abort()

  fetchingFlights = $.ajax('/flights', {
    data: { date: activeDiv },
    cache: false,

    error: function(result) {
      if (result.statusText !=
        $('#tabs #error').show();
      }
    }
  });
}
```

```
fetchingFlights = $.ajax '/flights'
  data:
    date: activeDiv
  cache: false
```

A Complex Example

```
showFlights = (activeDiv) ->
  $("#tabs div").hide()

  if fetchingFlights
    fetchingFlights.abort()

  fetchingFlights = $.ajax '/flights'
    data:
      date: activeDiv
    cache: false
```

```
    error: function(result) {
      if (result.statusText != "abort") {
        $('#tabs #error').show();
      }
    }
  });
}
```

```
error: (result) ->
  if result.statusText isnt "abort"
    $('#tabs #error').show()
```

A Complex Example

```
showFlights = (activeDiv) ->
  $("#tabs div").hide()

  if fetchingFlights
    fetchingFlights.abort()

  fetchingFlights = $.ajax '/flights'
    data:
      date: activeDiv
    cache: false
    error: (result) ->
      if result.statusText isnt "abort"
        $('#tabs #error').show()
```

46 Less Characters!

Mind Bending Comprehensions

```javascript
var filteredFlights = [];

$.each(currentFlights, function(index, flight) {
  if (stops == '2+' || flight.routing == 0) {
    filteredFlights.push(flight);
  }
});
```

JS

```coffeescript
filteredFlights = []

$.each currentFlights, (index, flight) ->
  if stops is '2+' or flight.routing is 0
    filteredFlights.push flight
```

Mind Bending Comprehensions

```
filteredFlights = []

$.each currentFlights, (index, flight) ->
  if stops is '2+' or flight.routing is 0
    filteredFlights.push flight
```

```
filteredFlights =
  (flight for flight in currentFlights when stops is '2+' or
                                            flight.routing is 0)
```

Applied jQuery, Part 2

• a sip of •

CoffeeScript

Object Orientation

• *level 6* •

Remember the Coffee Object?

```coffeescript
coffee =
   name: 'French'
   strength: 1
   brew: -> alert "brewing #{@name}"
   pour: (amount=1) ->
     if amount is 1
        "Poured a single cup"
     else
        "Poured #{amount} cups"
```

Constructors & Properties

```
class Coffee

 constructor: (name, strength=1) ->
   @name = name
   @strength = strength
```

called when instantiated

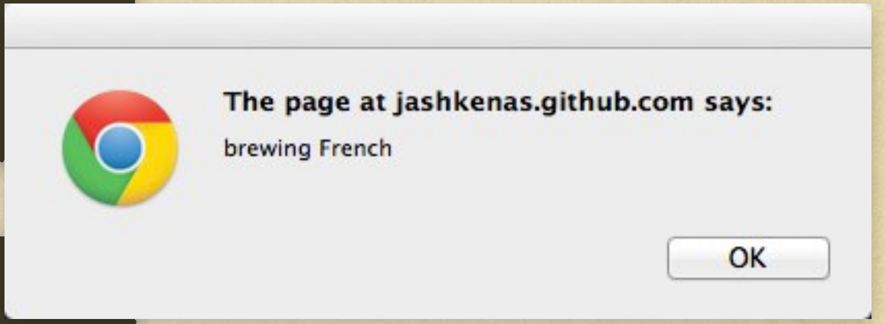
same as

```
class Coffee

 constructor: (@name, @strength=1) ->
```

Constructors & Properties

```
class Coffee

 constructor: (@name, @strength=1) ->

  brew: -> alert "brewing #{@name}"
  pour: (amount=1) ->
    if amount is 1
        "Poured a single cup"
    else
        "Poured #{amount} cups"
```

```
french = new Coffee("French", 2)

french.brew()
```


The page at jashkenas.github.com says:

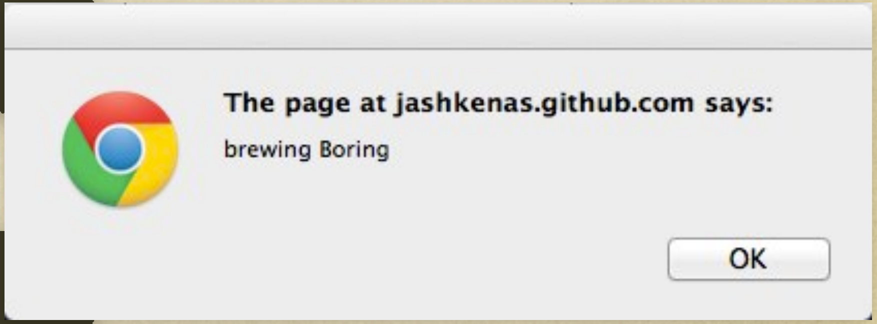
brewing French

Inheritance

```
class Coffee

 constructor: (@name, @strength=1) ->

 brew: -> alert "brewing #{@name}"
```

```
class MaxgoodHouse extends Coffee
  constructor: (@name, @strength=0) ->
    @brand = "Maxgood House"
```

```
boring = new MaxgoodHouse("Boring")

boring.brew()
```



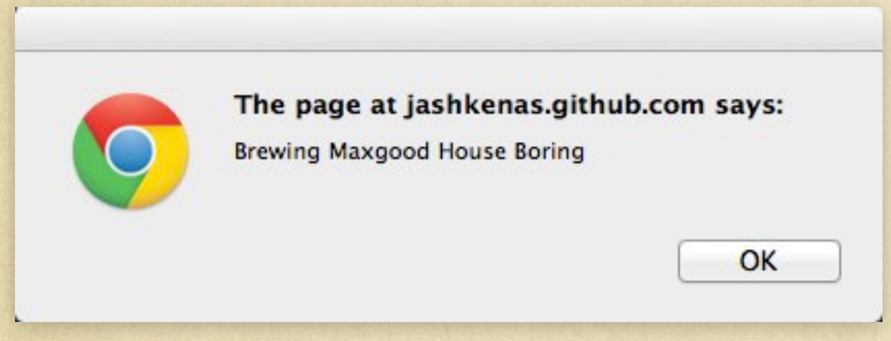
The page at jashkenas.github.com says:

brewing Boring

Inheritance

```
class MaxgoodHouse extends Coffee
  constructor: (@name, @strength=0) ->
    @brand = "Maxgood House"

  brew: -> alert "Brewing #{@brand} #{@name}"
```

```
boring = new MaxgoodHouse("Boring")

boring.brew()
```

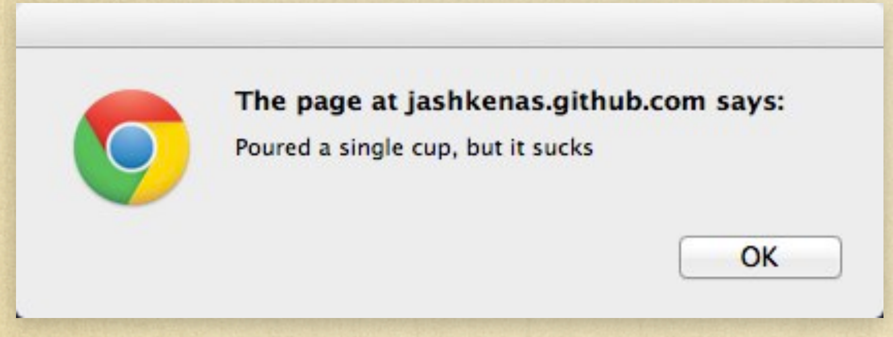

The page at jashkenas.github.com says:

Brewing Maxgood House Boring

Inheritance

```
class MaxgoodHouse extends Coffee
  constructor: (@name, @strength=0) ->
    @brand = "Maxgood House"

  brew: -> alert "Brewing #{@brand} #{@name}"
  pour: (amount=1) ->
    "#{super(amount)}, but it sucks"
```

```
boring = new MaxgoodHouse("Boring")

boring.pour()
```


The page at jashkenas.github.com says:

Poured a single cup, but it sucks

The Fat Arrow

```
class Coffee
  constructor: (@name, @strength=1, @inventory=0) ->

  pourClick: ->
    $("#pour-#{@name}").click (event) ->
      if @inventory isnt 0
        @inventory -= 1
        alert "Poured a cup of #{@name}"
```

✖ *Error!*

Looking for property of the dom element

called `@inventory` and `@name`

Object Orientation

The Fat Arrow

```
class Coffee
  constructor: (@name, @strength=1, @inventory=0) ->

  pourClick: ->
    $("#pour-#{@name}").click (event) =>
      if @inventory isnt 0
        @inventory -= 1
        alert "Poured a cup of #{@name}"
```

✓

Binds to current value of ‘this’

Using a Class for Encapsulation

```
var selectFlights = {
  fetchingFlights : null,

  init : function() {
    $("#tabs ul li a").bind({
      click: this.changeTab
    });

    $("#tabs #error a").click(function (event){
      e.preventDefault();
      this.showFlights($("#tabs li a.active").attr("href"));
    });
  },

  showFlights : function(activeDiv) { },
  changeTab : function(event) { }
});
```

```
class SelectFlights
```

JS

Object Orientation

CoffeeScript

Using a Class for Encapsulation

```
class SelectFlights
```

```
constructor: (@fetchingFlights=null) ->
```

```
  fetchingFlights : null,

  init : function() {
    $("#tabs ul li a").bind({
      click: this.changeTab
    });

    $("#tabs #error a").click(function (event){
      e.preventDefault();
      this.showFlights($("#tabs li a.active").attr("href"));
    });
  },

  showFlights : function(activeDiv) { },
  changeTab : function(event) { }
});
```

Object Orientation

CoffeeScript

Using a Class for Encapsulation

```
class SelectFlights
  constructor: (@fetchingFlights=null) ->
```

```
    $("#tabs ul li a").bind({
      click: this.changeTab
    });

    $("#tabs #error a").click(function (event){
      e.preventDefault();
      this.showFlights($("#tabs li a.active").attr("href"));
    });
  },

  showFlights : function(activeDiv) { },
  changeTab : function(event) { }
});
```

```
$("#tabs ul li a").bind
    click: @changeTab
```

Object Orientation

CoffeeScript

Using a Class for Encapsulation

```
class SelectFlights
  constructor: (@fetchingFlights=null) ->

    $("#tabs ul li a").bind
        click: @changeTab
```

```
    $("#tabs #error a").click(function (event){
      e.preventDefault();
      this.showFlights($("#tabs li a.active").attr("href"));
    });
  },
```

```
$("#tabs #error a").click (event) =>
  event.preventDefault()
  @showFlights $("#tabs li a.active").attr("href")
```

```
  showFlights : function(activeDiv) { },
  changeTab : function(event) { }
});
```

Object Orientation

Using a Class for Encapsulation

```
class SelectFlights
  constructor: (@fetchingFlights=null) ->

    $("#tabs ul li a").bind
        click: @changeTab

    $("#tabs #error a").click (event) =>
      event.preventDefault()
      @showFlights $("#tabs li a.active").attr("href")
```

```
  showFlights : function(activeDiv) { },
  changeTab : function(event) { }
});
```

```
showFlights : (activeDiv) ->
changeTab : (event) =>
```

Object Orientation

Using a Class for Encapsulation

```
class SelectFlights
  constructor: (@fetchingFlights=null) ->

    $("#tabs ul li a").bind
        click: @changeTab

    $("#tabs #error a").click (event) =>
      event.preventDefault()
      @showFlights $("#tabs li a.active").attr("href")

  showFlights : (activeDiv) ->
  changeTab : (event) =>
```

```
selectFlights = new SelectFlights()
```

• a sip of •

CoffeeScript