



PROGRAMMING FUNDAMENTALS DAY 3

Andrew Buntine
Technical Director, Hardhat

DEVELOPMENT ENVIRONMENT SETUP

Create a Nitrous account (for authoring)

<https://nitrous.io/>

Create a Github account (for collaborating)

<https://github.com/>

New Box

Pick a template



Ruby/Rails



Node.js



Python/Django



Go



PHP

Name

Region

Memory **384 MB** 120

Storage **1000 MB** 20

Remaining N2O **20** 140

Download a Github repo

Create Box

Cancel

AGENDA

LEVEL 1

- VOCABULARY
- PROCESS
- BASICS OF CODE

LEVEL 2

- HTML
- CSS
- Javascript

LEVEL 3

- RUBY/RAILS
- CREATE A BASIC WEB APPLICATION

LEVEL 4

- PUTTING IT ALL TOGETHER



Recap

Development Environment

What is backend programming?

Thinking like a programmer

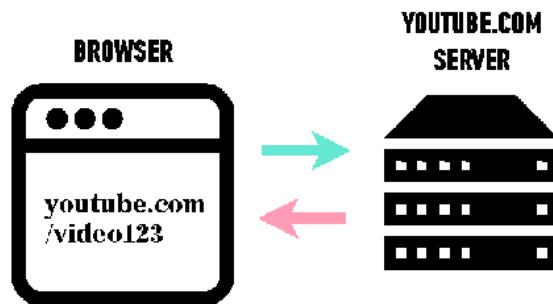
Ruby

Backend frameworks



RECAP

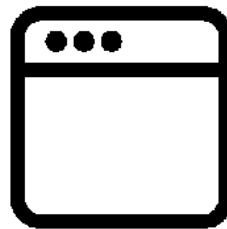
HOW THE WEB WORKS



BROWSER REQUEST: "Connect me to server youtube.com. Great, please send me /video123"

SERVER RESPONSE: "Okay. I have found /video123, it is a HTML file. Here is the HTML."

FRONT-END AND BACK-END



FRONT-END

HTML, CSS
Javascript

BACK-END

Ruby, Python
PHP, Databases

HTML “DOCUMENT”

```
<!doctype html>
<html>
  <head>
    <title>Hipster Coffee Co.</title>
  </head>
  <body>
    <p>Tasty coffee</p>
  </body>
</html>
```

TAGS

Root element	<code><html></code>
Document metadata	<code><head> <title> <link> <meta></code>
Scripting	<code><script></code>
Sections	<code><section> <nav> <article> <header> <footer></code>
Grouping	<code><p> <dl> <div></code>
Text	<code><a> </code>
Embedded content	<code> <iframe> <video> <audio> <canvas> <svg></code>
Tabular data	<code><table> <tr> <th> <td></code>
Forms	<code><form> <fieldset> <input> <label></code>

TRAVEL BLOG HTML

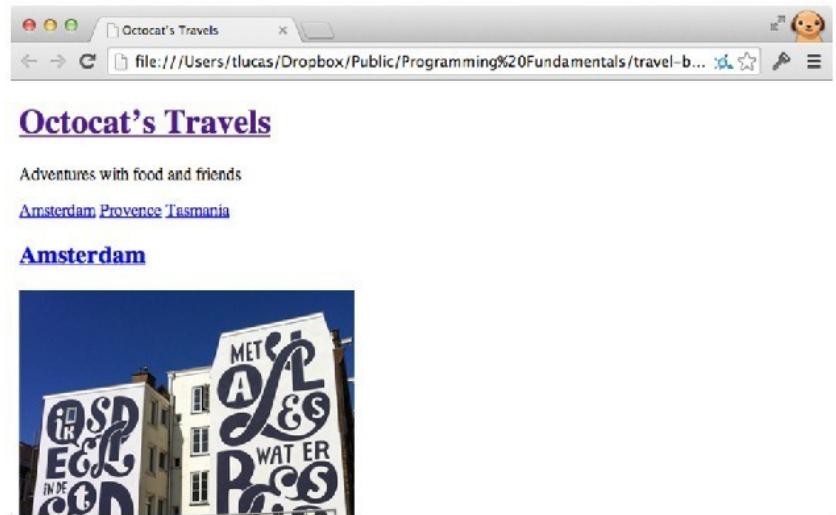
HTML structure

Tags

Attributes

Chrome Dev Tools

Sections (header, footer, section, etc)

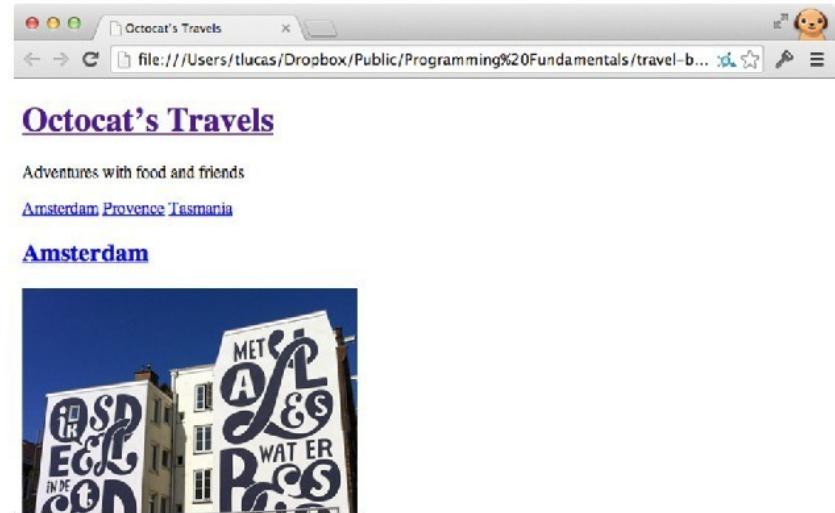


USER AGENT STYLESHEET

Built into every browser

The default stylesheet

See them with Chrome Dev Tools



NORMALIZE VS RESET

normalize.css

- Makes all browsers look the same
- Retains default styles

reset.css

- Removes all the default styles
- Need to make sure you re-add your own styles

CSS SYNTAX

```
selector {  
  property: value;  
}
```

CSS PROPERTIES

```
.site-footer {  
    font: bold 21px "Comic Sans MS", sans-serif;  
    border: 1px solid red;  
}
```

JAVASCRIPT

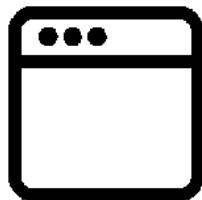
```
var a = document.querySelector("a");

a.addEventListener("click", function() {
  var name = prompt("Hi, I'm T-Rex. Who are you?");
  a.textContent = "Hai " + name + "! :)";
});
```

JQUERY

```
$("a").click(function() {  
    var name = prompt("Hi, I'm T-Rex. Who are you?");  
    $(this).text("Hai " + name + "! :)");  
});
```

WHAT IS A LIBRARY?



FRONT END

Platform: Browser
Language: Javascript
Libraries: **jQuery**
 jQuery.UI
 Backbone
 Ember



BACK END

Platform: Node.js
Language: Javascript
Libraries: **mocha**
 express
 connect
 request
(Modules)

IMAGE FORMATS

SVG + JPEG COVERS 99% OF NEEDS

SVG for all logos, icons etc

JPEG for all photograph(bitmap images

Future proof + size friendly

BACKEND DEVELOPMENT

DEVELOPMENT ENVIRONMENT SETUP

Create a Nitrous account (for authoring)

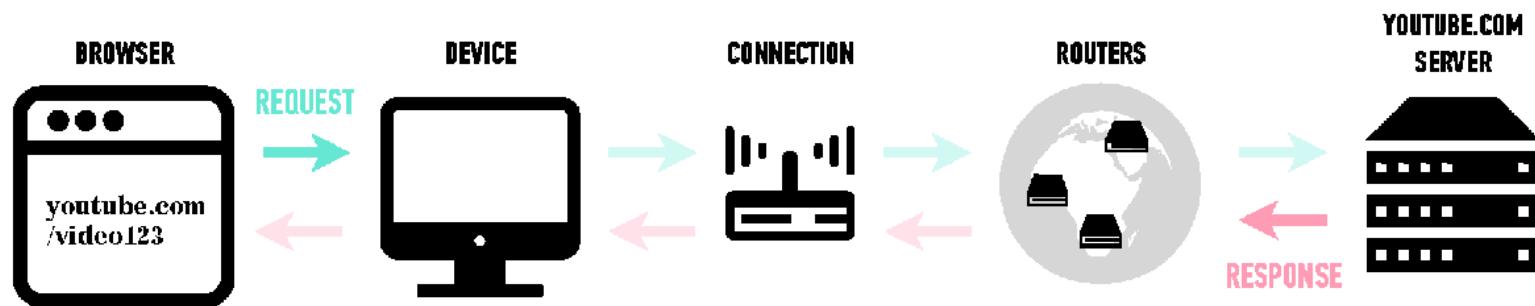
<https://nitrous.io/>

Create a Github account (for collaborating)

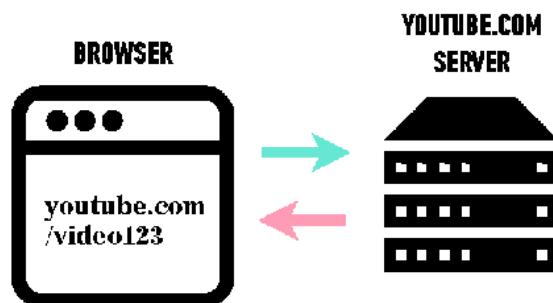
<https://github.com/>

WHAT IS BACKEND DEVELOPMENT?

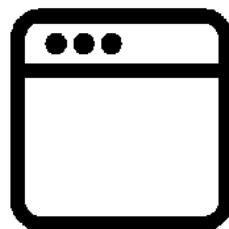
HOW THE WEB WORKS



HOW THE WEB WORKS



HOW THE WEB WORKS



FRONT-END

HTML, CSS
Javascript

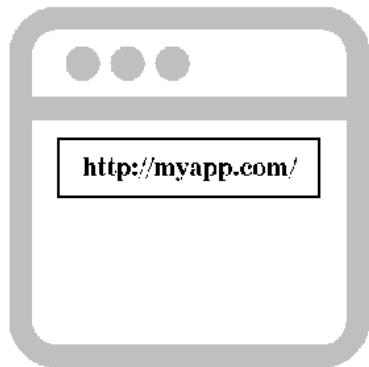


BACK-END

Ruby, Python
PHP, Databases

WHAT IS BACKEND DEVELOPMENT?

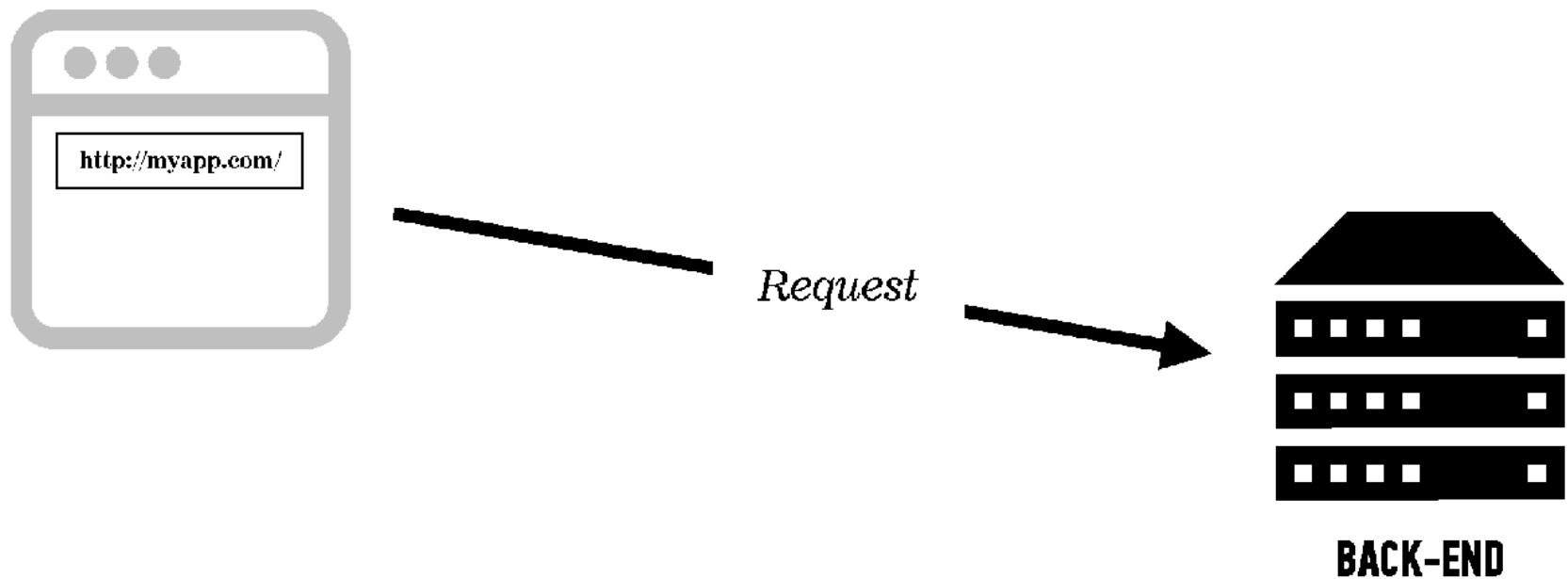
29



BACK-END

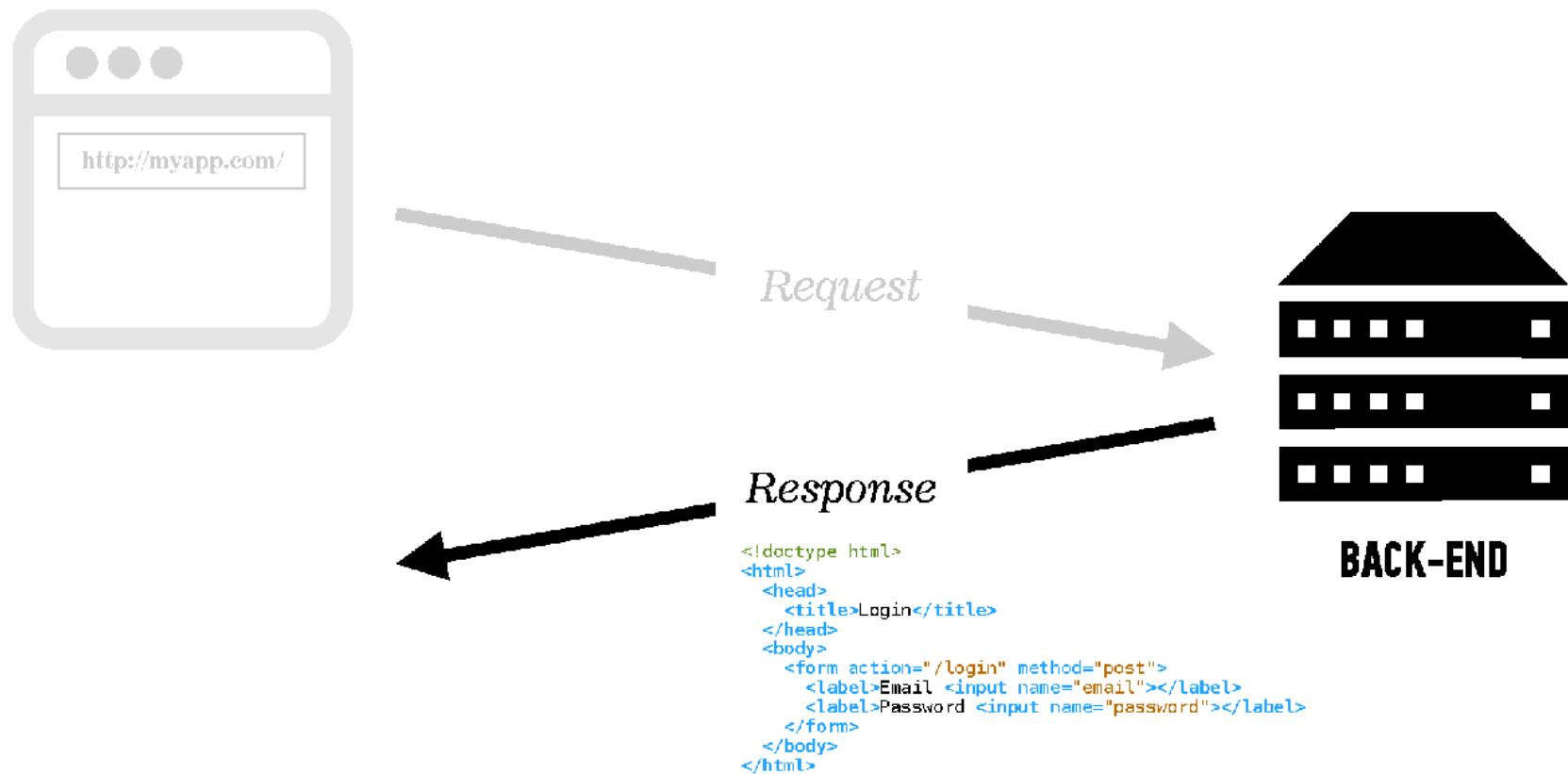
WHAT IS BACKEND DEVELOPMENT?

30



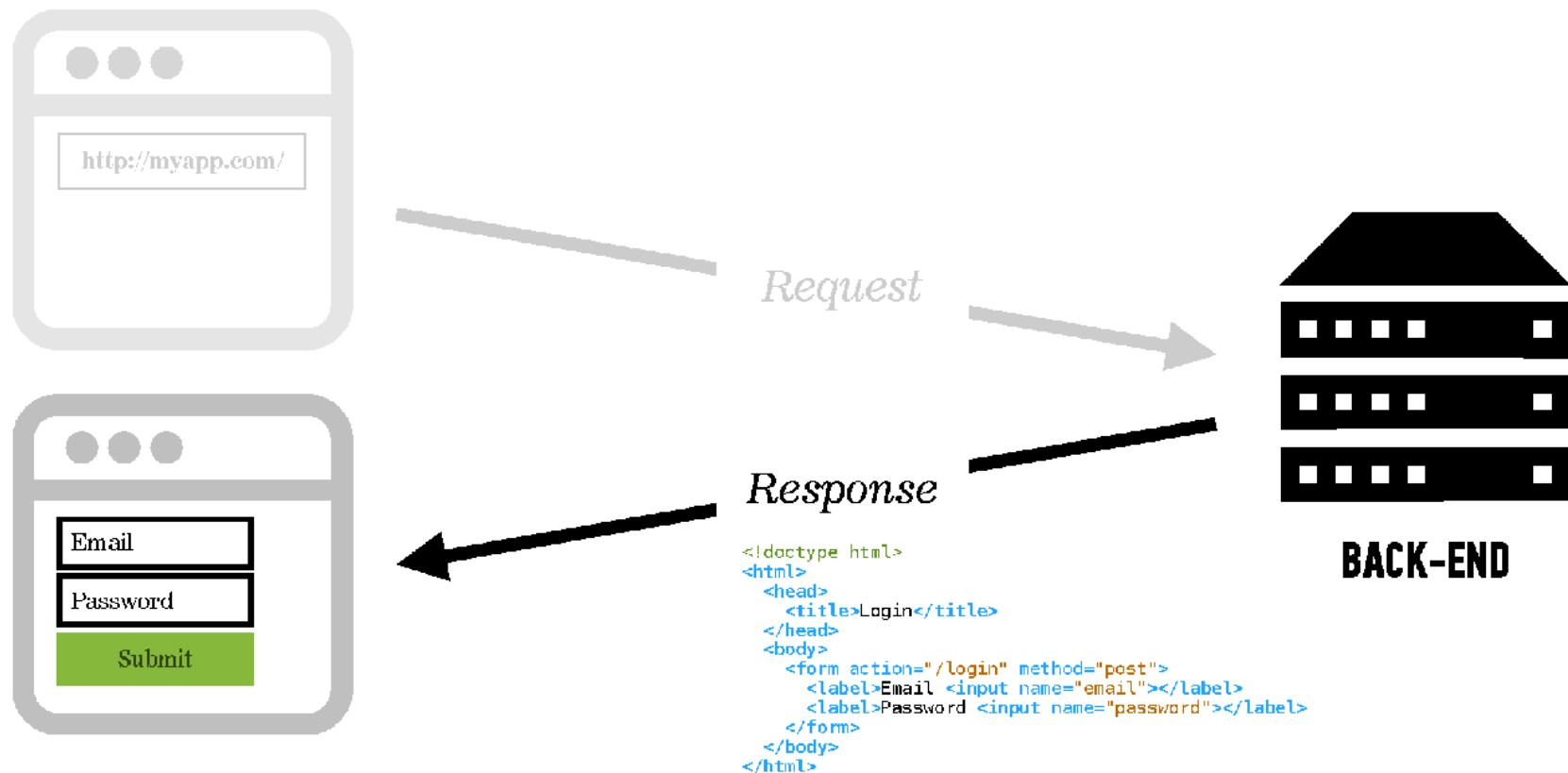
WHAT IS BACKEND DEVELOPMENT?

31



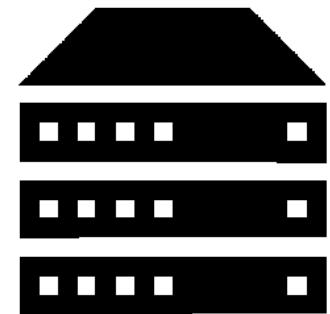
WHAT IS BACKEND DEVELOPMENT?

32



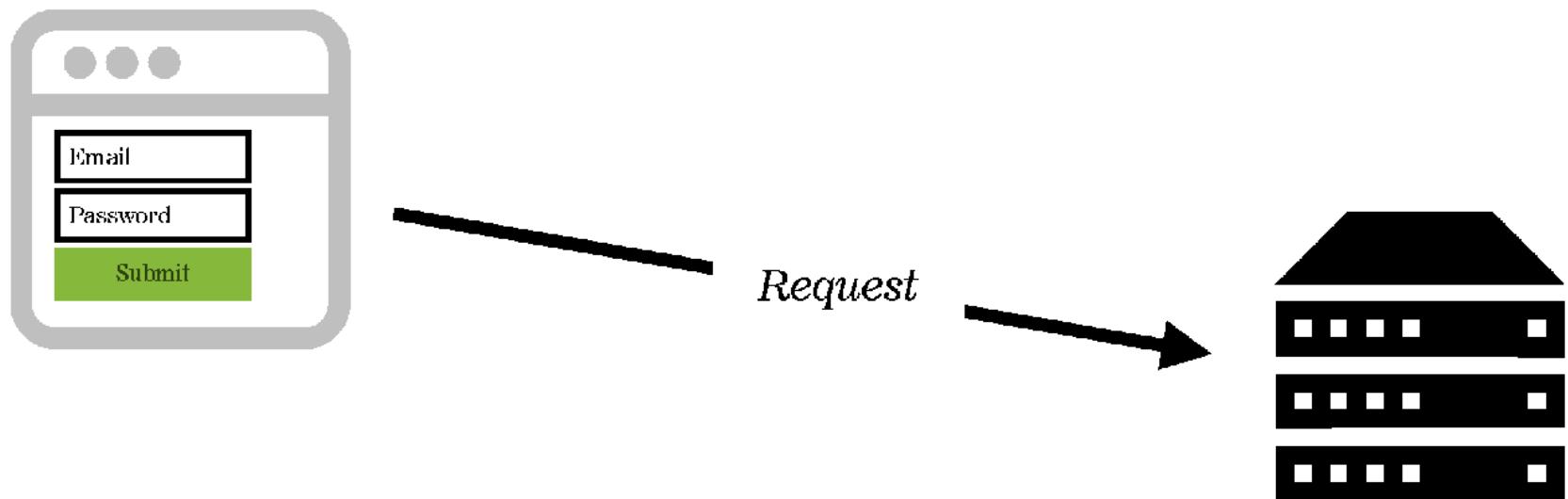
WHAT IS BACKEND DEVELOPMENT?

33



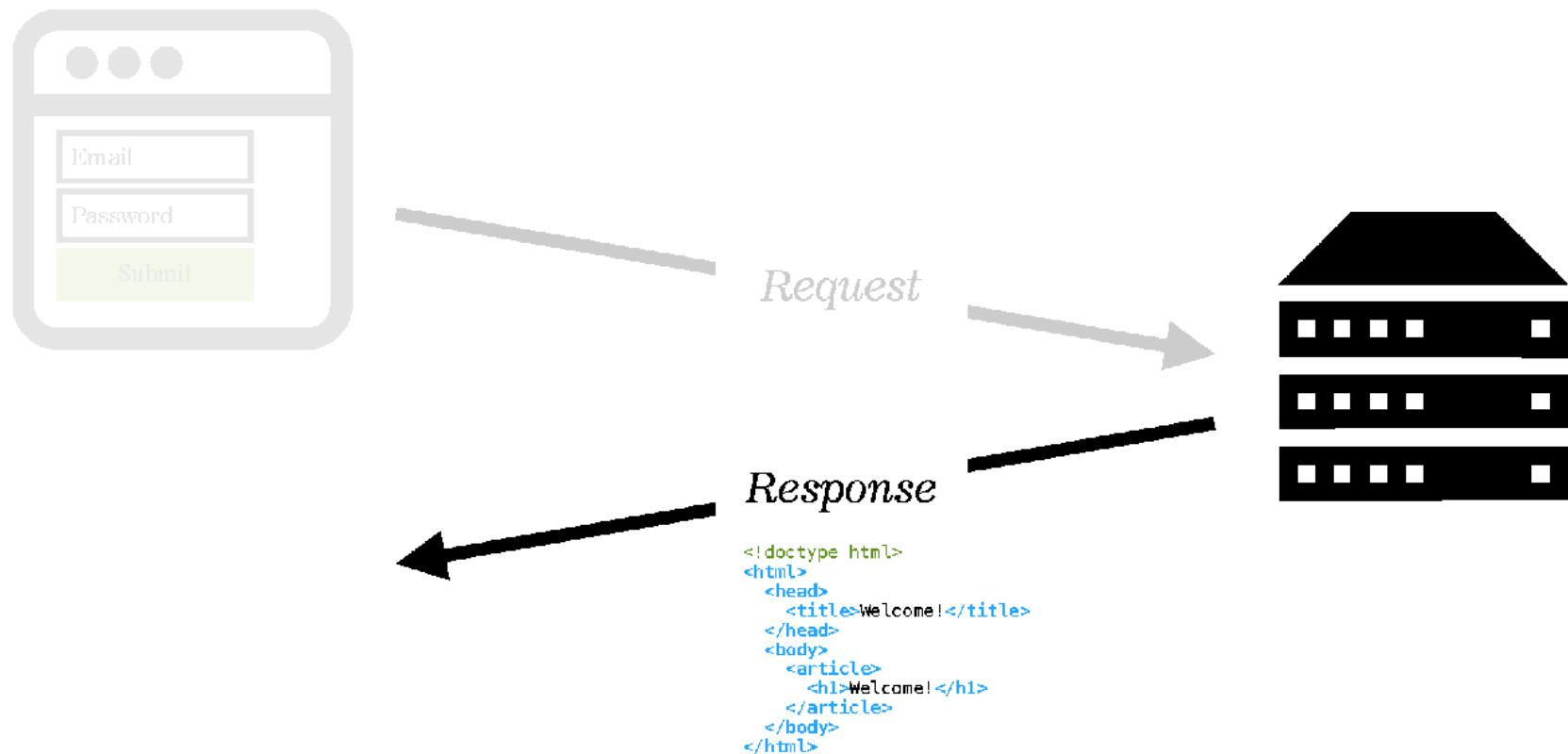
WHAT IS BACKEND DEVELOPMENT?

34



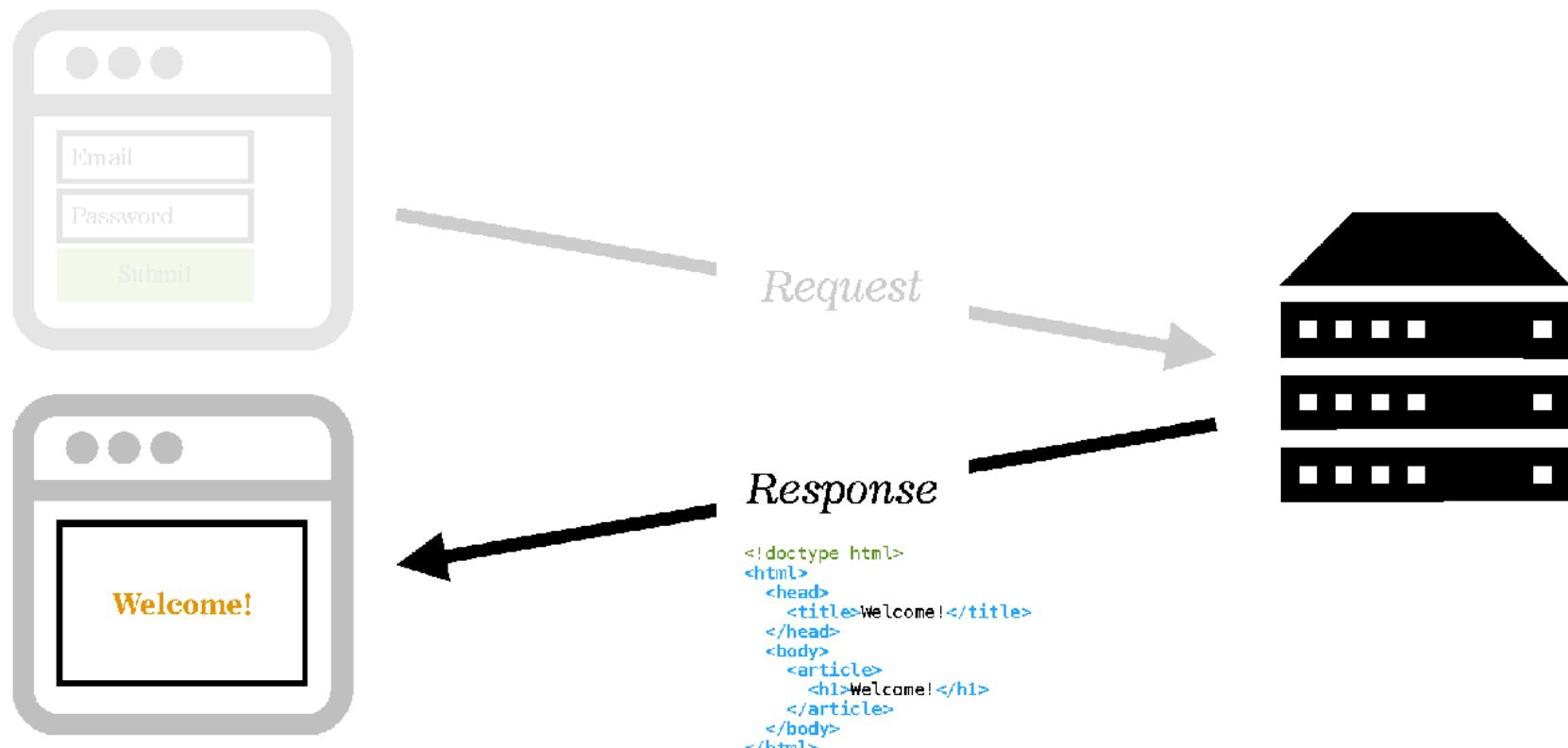
WHAT IS BACKEND DEVELOPMENT?

35



WHAT IS BACKEND DEVELOPMENT?

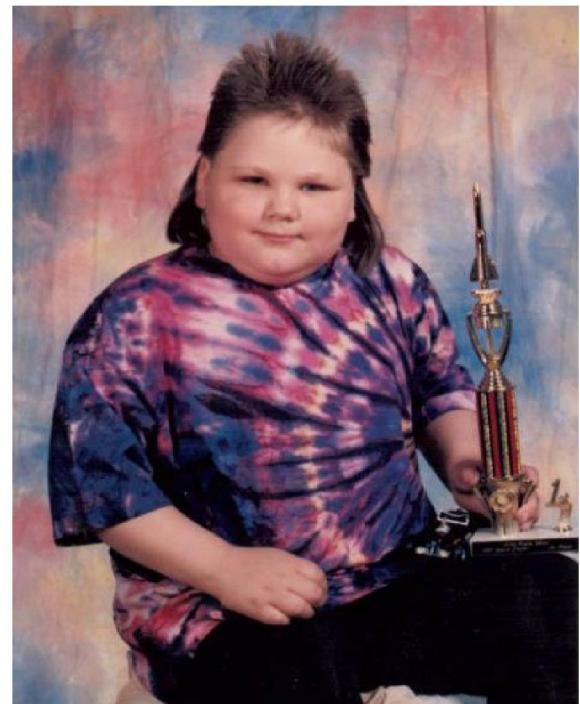
36



WHAT ELSE CAN YOU DO?

Use APIs

- *Grab a picture from Reddit.com using their API*
- *Email that picture to a friend using the Mandrill API*



DEMO

<http://picster.hhd.com.au>

Source code:

<http://github.com/buntine/picster>

WHAT ELSE CAN YOU DO?

Retrieve, store and update information from a database

Read, store and update files

Render templates

Set HTTP cookies

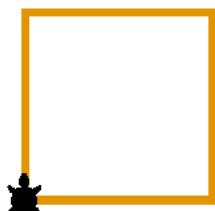
PROGRAMMING CONCEPTS

EXERCISE: TURTLE

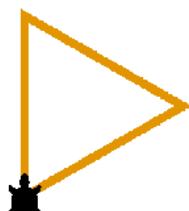
Imagine there was a turtle that understood the instructions (functions) on the right.

Break up into *groups of 2*

Write down a recipe (routine) for each of the pictures below, using the functions available.



Square



Triangle



Double Square

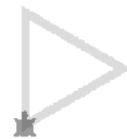
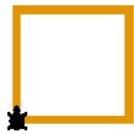
Documentation

Available turtle functions

- Forward x metres
- Turn right x degrees
- Turn left x degrees
- Repeat x times [
 - *instructions*]

PROGRAMMING CONCEPTS: TURTLE SOLUTIONS

43

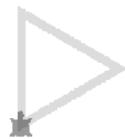


PROGRAMMING CONCEPTS: TURTLE SOLUTIONS

44



```
Forward 5 metres  
Turn right 90 degrees  
Forward 5 metres  
Turn right 90 degrees  
Forward 5 metres  
Turn right 90 degrees  
Forward 5 metres  
Turn right 90 degrees
```



PROGRAMMING CONCEPTS: TURTLE SOLUTIONS

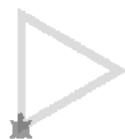
45



Forward 5 metres
Turn right 90 degrees
Forward 5 metres
Turn right 90 degrees
Forward 5 metres
Turn right 90 degrees
Forward 5 metres
Turn right 90 degrees

or

```
Repeat 4 times [  
  Forward 5 metres  
  Turn right 90 degrees  
]
```



PROGRAMMING CONCEPTS: TURTLE SOLUTIONS

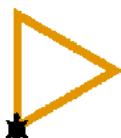
46



```
Forward 5 metres  
Turn right 90 degrees  
Forward 5 metres  
Turn right 90 degrees  
Forward 5 metres  
Turn right 90 degrees  
Forward 5 metres  
Turn right 90 degrees
```

or

```
Repeat 4 times [  
  Forward 5 metres  
  Turn right 90 degrees  
]
```



PROGRAMMING CONCEPTS: TURTLE SOLUTIONS

47



Forward 5 metres
Turn right 90 degrees
Forward 5 metres
Turn right 90 degrees
Forward 5 metres
Turn right 90 degrees
Forward 5 metres
Turn right 90 degrees

or

```
Repeat 4 times [  
  Forward 5 metres  
  Turn right 90 degrees  
]
```



Forward 5 metres
Turn right 120 degrees
Forward 5 metres
Turn right 120 degrees
Forward 5 metres
Turn right 120 degrees



PROGRAMMING CONCEPTS: TURTLE SOLUTIONS

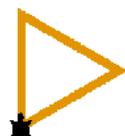
48



Forward 5 metres
Turn right 90 degrees
Forward 5 metres
Turn right 90 degrees
Forward 5 metres
Turn right 90 degrees
Forward 5 metres
Turn right 90 degrees

or

```
Repeat 4 times [  
  Forward 5 metres  
  Turn right 90 degrees  
]
```



Forward 5 metres
Turn right 120 degrees
Forward 5 metres
Turn right 120 degrees
Forward 5 metres
Turn right 120 degrees

or

?



PROGRAMMING CONCEPTS: TURTLE SOLUTIONS

49



Forward 5 metres
Turn right 90 degrees
Forward 5 metres
Turn right 90 degrees
Forward 5 metres
Turn right 90 degrees
Forward 5 metres
Turn right 90 degrees

or

Repeat 4 times [
 Forward 5 metres
 Turn right 90 degrees
]



Forward 5 metres
Turn right 120 degrees
Forward 5 metres
Turn right 120 degrees
Forward 5 metres
Turn right 120 degrees

or

Repeat 3 times [
 Forward 5 metres
 Turn right 120 degrees
]



PROGRAMMING CONCEPTS: TURTLE SOLUTIONS

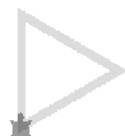
50



Forward 5 metres
Turn right 90 degrees
Forward 5 metres
Turn right 90 degrees
Forward 5 metres
Turn right 90 degrees
Forward 5 metres
Turn right 90 degrees

or

```
Repeat 4 times [  
  Forward 5 metres  
  Turn right 90 degrees  
]
```



Forward 5 metres
Turn right 120 degrees
Forward 5 metres
Turn right 120 degrees
Forward 5 metres
Turn right 120 degrees

or

```
Repeat 3 times [  
  Forward 5 metres  
  Turn right 120 degrees  
]
```



PROGRAMMING CONCEPTS: TURTLE SOLUTIONS

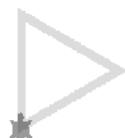
51



Forward 5 metres
Turn right 90 degrees
Forward 5 metres
Turn right 90 degrees

or

Repeat 4 times [
 Forward 5 metres
 Turn right 90 degrees
]



Forward 5 metres
Turn right 120 degrees
Forward 5 metres
Turn right 120 degrees
Forward 5 metres
Turn right 120 degrees
Forward 5 metres
Turn right 120 degrees

or

Repeat 3 times [
 Forward 5 metres
 Turn right 120 degrees
]



Repeat 4 times [
 Forward 5 metres
 Turn right 90 degrees
]

PROGRAMMING CONCEPTS: TURTLE SOLUTIONS

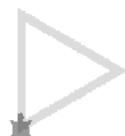
52



Forward 5 metres
Turn right 90 degrees
Forward 5 metres
Turn right 90 degrees

or

Repeat 4 times [
 Forward 5 metres
 Turn right 90 degrees
]



Forward 5 metres
Turn right 120 degrees
Forward 5 metres
Turn right 120 degrees
Forward 5 metres
Turn right 120 degrees
Forward 5 metres
Turn right 120 degrees

or

Repeat 3 times [
 Forward 5 metres
 Turn right 120 degrees
]



Repeat 4 times [
 Forward 5 metres
 Turn right 90 degrees
]
Turn right 45 degrees

PROGRAMMING CONCEPTS: TURTLE SOLUTIONS

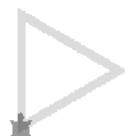
53



Forward 5 metres
Turn right 90 degrees
Forward 5 metres
Turn right 90 degrees

or

```
Repeat 4 times [  
  Forward 5 metres  
  Turn right 90 degrees  
]
```



Forward 5 metres
Turn right 120 degrees
Forward 5 metres
Turn right 120 degrees
Forward 5 metres
Turn right 120 degrees
or

```
Repeat 3 times [  
  Forward 5 metres  
  Turn right 120 degrees  
]
```



```
Repeat 4 times [  
  Forward 5 metres  
  Turn right 90 degrees  
]  
Turn right 45 degrees  
Repeat 4 times [  
  Forward 5 metres  
  Turn right 90 degrees  
]
```

PROGRAMMING CONCEPTS: TURTLE SOLUTIONS

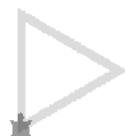
54



Forward 5 metres
Turn right 90 degrees
Forward 5 metres
Turn right 90 degrees

or

Repeat 4 times [
 Forward 5 metres
 Turn right 90 degrees
]



Forward 5 metres
Turn right 120 degrees
Forward 5 metres
Turn right 120 degrees
Forward 5 metres
Turn right 120 degrees
or

Repeat 3 times [
 Forward 5 metres
 Turn right 120 degrees
]



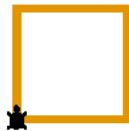
Repeat 4 times [
 Forward 5 metres
 Turn right 90 degrees
]
Turn right 45 degrees
Repeat 4 times [
 Forward 5 metres
 Turn right 90 degrees
]

or

Repeat 2 times [
 Repeat 4 times [
 Forward 5 metres
 Turn right 90 degrees
]
 Turn right 45 degrees
]

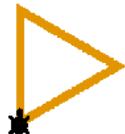
ALGORITHMS

AKA FUNCTIONS, METHODS, ROUTINES



Square

```
Repeat 4 times [  
  Forward 5 metres  
  Turn right 90 degrees  
]
```



Triangle

```
Repeat 3 times [  
  Forward 5 metres  
  Turn right 120 degrees  
]
```



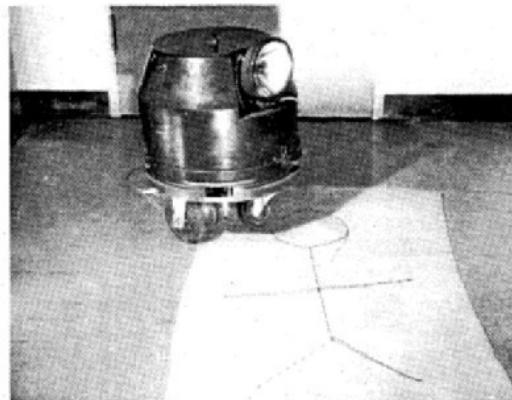
Double Square

```
Repeat 2 times [  
  Repeat 4 times [  
    Forward 5 metres  
    Turn right 90 degrees  
  ]  
  Turn right 45 degrees  
]
```

YOU JUST WROTE LOGO

<http://www.calormen.com/jslogo/>

1. Make a Turtle



The picture shows one of our turtles . . . so-called in honor of a famous species of cybernetic animal made by Grey Walter, an English neurophysiologist. Grey Walter's turtles had life-like behavior patterns built into its wiring diagram. Ours have no behavior except the ability to obey a few simple commands from a computer to which they are attached by a wire that plugs into a control-box that connects to a telephone line that speaks to the computer, which thinks it is talking to a

PERSON

FROM "TWENTY THINGS TO DO WITH A COMPUTER" 1971, SEYMOUR PAPERT & CYNTHIA SOLOMON

```
TO draw :distance
  forward :distance
  back :distance
END
```

```
TO vee :size
  left 50
  draw :size
  right 100
  draw :size
  left 50
END
```

```
TO circle :diameter
  arc 360 :diameter/2
END
```

```
TO person :size
  right 180
  vee :size
  right 180
  forward :size
  vee :size
  forward :size/2
  penup
  forward 50
  pendown
  circle :size
END
```

```
person 100
person 10
```



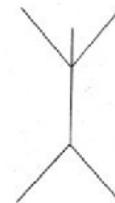
The picture shows one of our turtles . . . so-called in honor of a famous species of eiderdown animal made by Grey Walter, an English neurophysiologist. Grey Walter's turtles had life-like behavior patterns built into the wiring diagram. Ours have no behavior except the ability to obey a few simple commands from a computer to which they are attached by a wire that plugs into a control-box that connects to a telephone line that speaks to the computer, which thinks it is talking to a

```
Now
TO MAN :SIZE
1 VEE :SIZE ←
2 RIGHT 180
3 FORWARD :SIZE
4 VEE :SIZE
5 FORWARD :SIZE/2
```

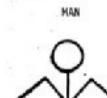
MAN 100 will draw

MAN 10 will draw

We now use the previously defined command in making our new command. In other words TO DRAW was a sub-procedure of TO VEE; TO VEE was a sub-procedure of TO MAN.



Here are some other drawings the fifth grade kids made the turtle draw.



A CREATIVE PROCESS

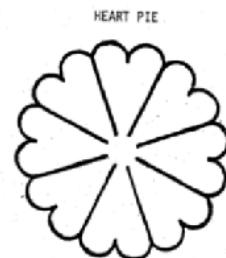
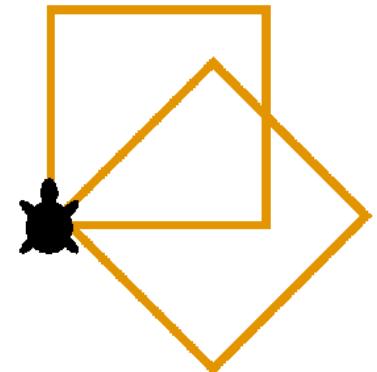
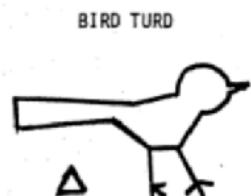
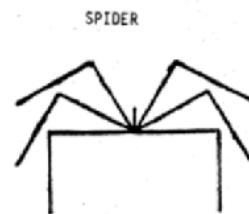
Understanding the problem

Exploring solutions

Pattern recognition

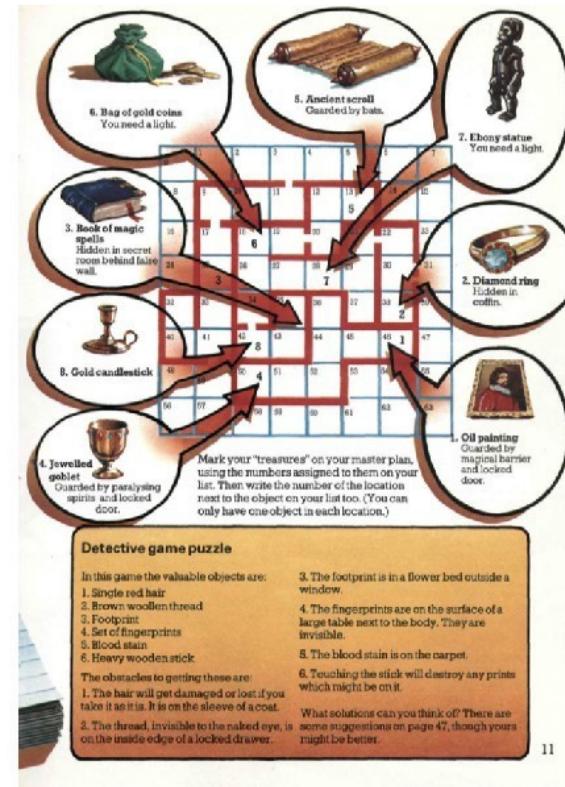
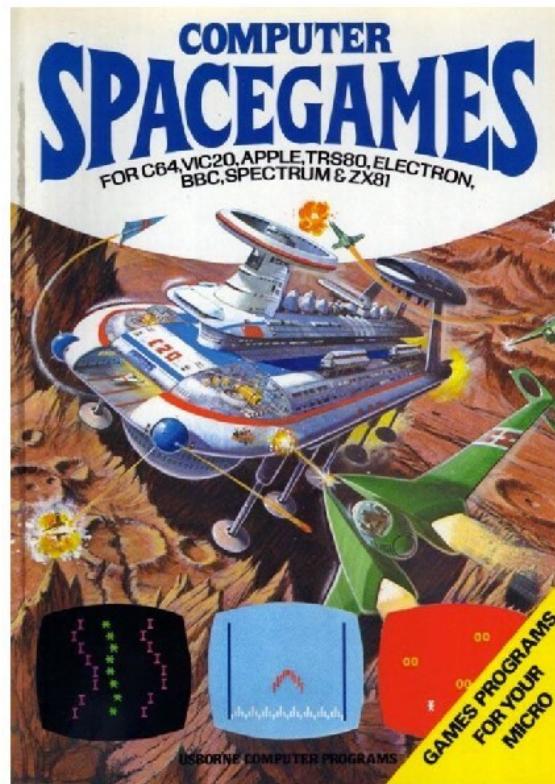
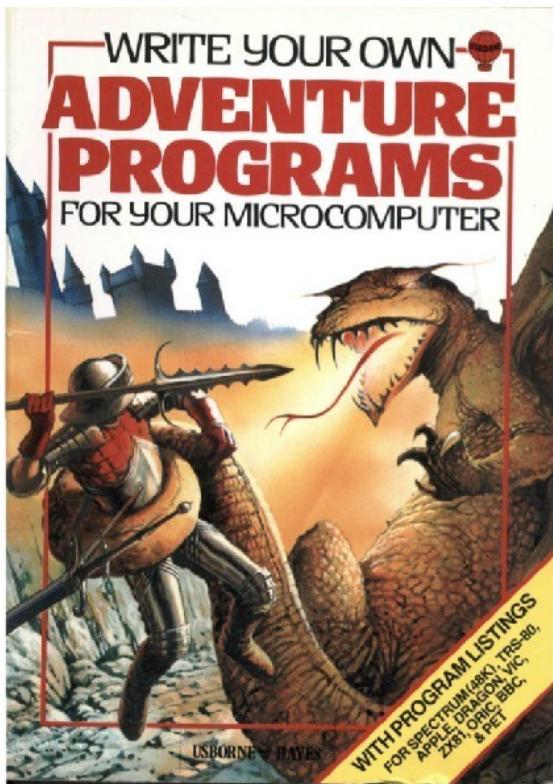
Abstraction

Sharing ideas



PROGRAMMING CONCEPTS

59



IDEAS JUST COVERED

Functions and libraries

Pre-built library functions such as Forward

Looping

Repeat things using repeat

Abstraction

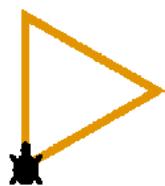
Created our own functions such as square

Parameters

Allowed for adjustments of numbers such as length

ABSTRACTION

Problem



Solution

```
repeat 3 [  
    forward 100  
    right 120  
]
```

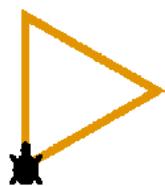
Abstraction

```
T0 triangle  
repeat 3 [  
    forward 100  
    right 120  
]  
END
```

Re-Use

ABSTRACTION

Problem



Solution

```
repeat 3 [  
    forward 100  
    right 120  
]
```

Abstraction

```
T0 triangle  
repeat 3 [  
    forward 100  
    right 120  
]  
END
```

Re-Use

```
left 90  
repeat 2 [  
    triangle  
    forward 25  
]
```



WHY ABSTRACT YOUR CODE?

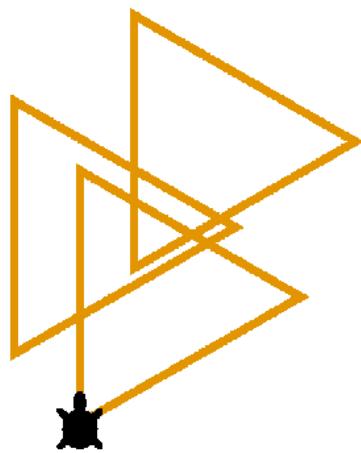
- Simplifies your code
- Allows yourself (and others) to re-use your work
- Libraries and frameworks provide pre-built abstractions



PARAMETERS

Without parameters

```
T0 triangle  
repeat 3 [  
    forward 100  
    right 120  
]  
END
```

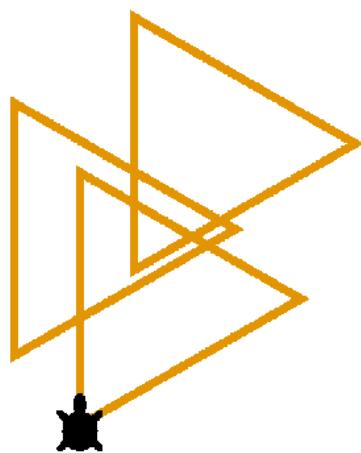


With parameters

PARAMETERS

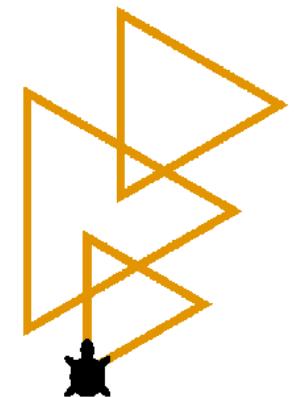
Without parameters

```
T0 triangle  
repeat 3 [  
    forward 100  
    right 120  
]  
END
```



With parameters

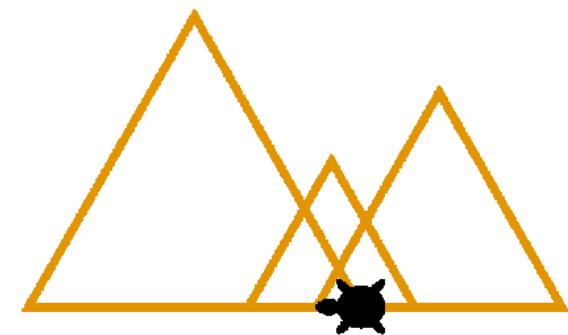
```
T0 triangle :size  
repeat 3 [  
    forward :size  
    right 120  
]  
END
```



WHY USE PARAMETERS?

Benefits:

- Makes your code easier to read
- Makes your code more re-usable and generic

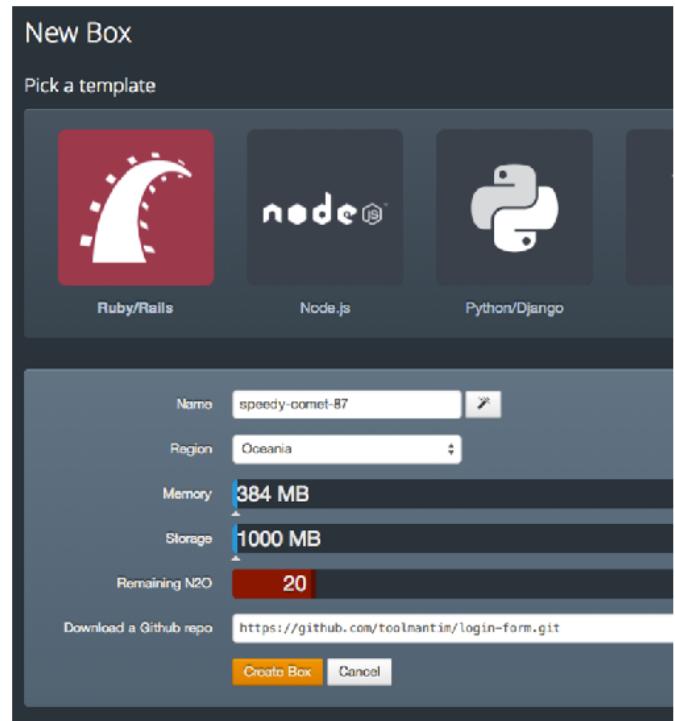


```
T0 triangle :size
repeat 3 [
    forward :size
    right 120
]
END
left 90
triangle 80
forward 50
triangle 50
forward 20
triangle 100
```

PROGRAMMING CONCEPTS (USING RUBY)

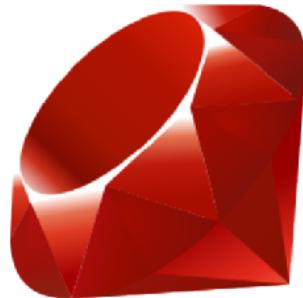
FIRST RUBY PROGRAM

- Open <https://nitrous.io/>
- Create a new Rails box
- Create a new file `hello.rb` inside Workspace with:
`puts "Hello world"`
- Save the file
- In the console:
`ruby hello.rb`



WHAT IS RUBY?

A language designed and developed in 1993
by Yukhiro Matsumoto (aka “Matz”)



"I hope to see Ruby help every programmer in the world to be productive, and to enjoy programming, and to be happy. That is the primary purpose of Ruby language."

VARIABLES

STORING THINGS

```
name = "Sally"  
puts name # this will print "Sally"
```

```
age = 2013 - 1993  
puts age # what will this print?
```



BASIC DATA TYPES

```
puts "Sally"
```

```
puts 20
```

```
puts [1,2,3]
```

```
"Sally" + [1,2,3] # won't work
```

CONDITIONALS

```
if age < 12
  puts "Child"
elsif age > 12 and age < 20
  puts "Teenager"
else
  puts "Adult"
end
```

CONDITIONALS

```
if name == "Sally"
  puts "User is Sally"
else
  puts "User is not Sally"
end
```

COMPARISON OPERATORS

Operator	Description	Example (a =4 and b= 2)
<code>==</code>	Equal	<code>a == b</code> <code>false</code>
<code>!=</code>	Not Equal	<code>a != b</code> <code>true</code>
<code>></code>	Greater than	<code>a > b</code> <code>true</code>
<code><</code>	Less than	<code>a < b</code> <code>true</code>
<code>>=</code>	Greater than or equal to	<code>a <= b</code> <code>false</code>
<code><=</code>	Less than or equal to	<code>a <= b</code> <code>false</code>
<code> <=> </code>	<code>same value? return 0</code> <code>less than? return -1</code> <code>greater than? return 1</code>	<code>a <=> b</code> <code>1</code>
<code>.eq?</code>	same value and same type?	<code>1.eq?(1.0)</code> <code>false</code>

LOOPS

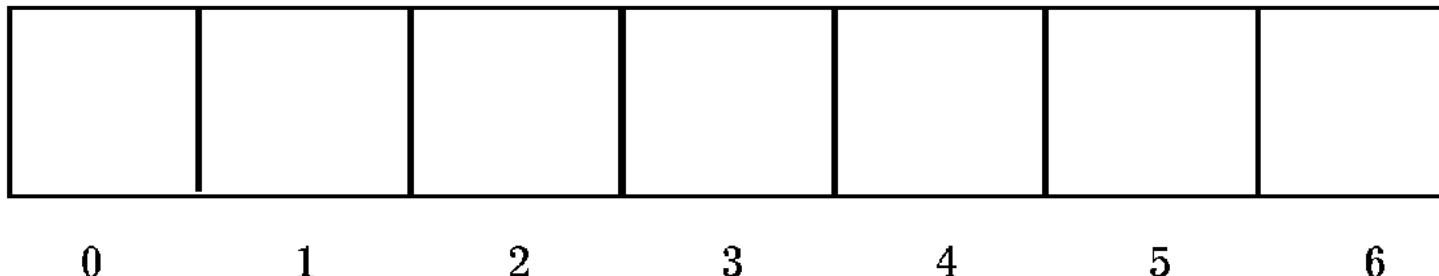
REPEATING THINGS

Print your name 10 times

```
10.times do
  puts "Don"
end
```

ARRAYS

STORING MULTIPLE THINGS



ARRAYS

STORING MULTIPLE THINGS

```
# Store 3 names
names = ["Don", "Sally", "Roger"]

# Print all the names
names.each do |name|
    puts name
end
```

ARRAYS

STORING MULTIPLE THINGS

```
# Store 3 names
names = ["Don", "Sally", "Roger"]

# Print all the names (with indices)
names.each_with_index do |name, index|
  puts "#{index}: #{name}"
end
```

FUNCTIONS

DEFINING YOUR OWN

```
# Define a function called "say_hello"
def say_hello_to(name)
    puts "Hello #{name}"
end

say_hello_to "Don"    # Prints "Hello Don"
say_hello_to "Sally" # Prints "Hello Sally"
say_hello_to "Roger" # Prints "Hello Roger"
```

LEARNING RUBY

Interactive web learning: <http://dash.generalassembly.com/>

Another interactive web learning: <http://tryruby.org/>

Why's Poignant Guide to Ruby: <http://mislav.uniqpath.com/poignant-guide/book/>

Learn to program with Ruby: <http://pragprog.com/book/ltp2/learn-to-program>

READING DOCUMENTATION

For built-in Ruby methods: <http://ruby-doc.org/>

For Ruby gems: <http://rubydoc.info/>

Reference book: <http://pragprog.com/book/ruby4/programming-ruby-1-9-2-0>

WEB FRAMEWORKS

WHAT IS A WEB FRAMEWORK?

Toolkit for building web applications

Has library functions for handling:

- Requests
- Responses
- Templates
- Cookies

RUBY WEB FRAMEWORKS

Two main ones in Ruby:

- Rails
Kitchen sink, pre-built
- Sinatra
Minimal, put it together yourself

SINATRA

Sinatra has the concept of:

- Routes
- Views (templates)

You add the rest yourself.

RAILS

Rails has the concept of:

- Routes
- Controllers
- Views (templates)
- Models (database)
- Mailers (email)
- and more...

LIBRARIES ARE PACKAGED AS “GEMS”



Rails
Sinatra
Devise
Carrier Wave
Geocoder

“There’s a gem for that”

HOW TO INSTALL GEMS

```
$ gem install rails  
$ gem install geocoder
```

BUNDLER

You normally put a list of all the gems you use for your project in a Gemfile, and use Bundler to install all the gems for you.

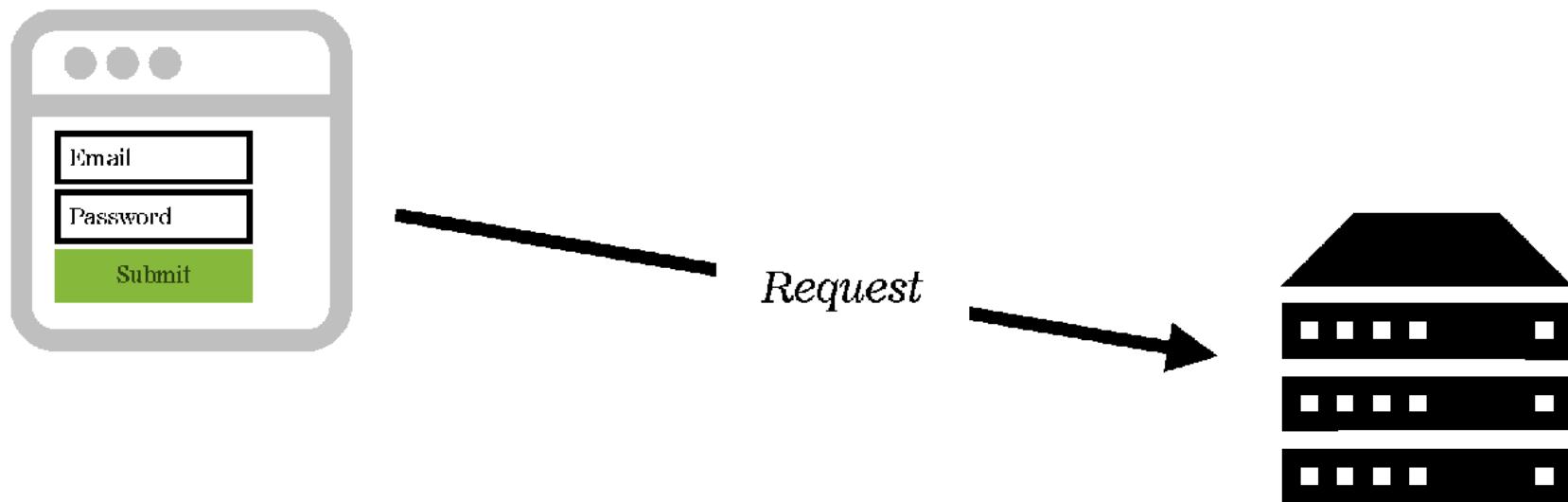
```
$ bundle install
```

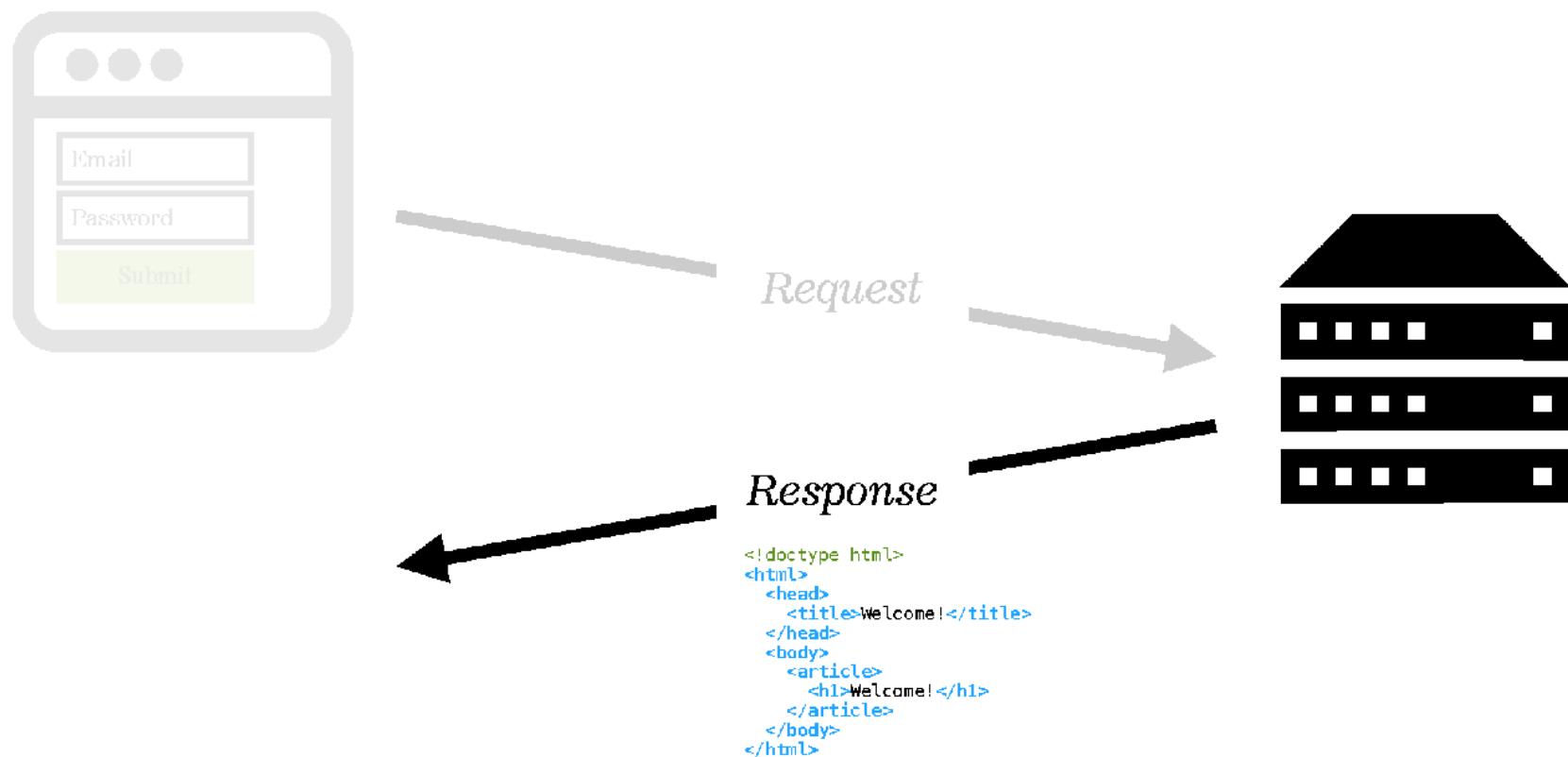
RECAP

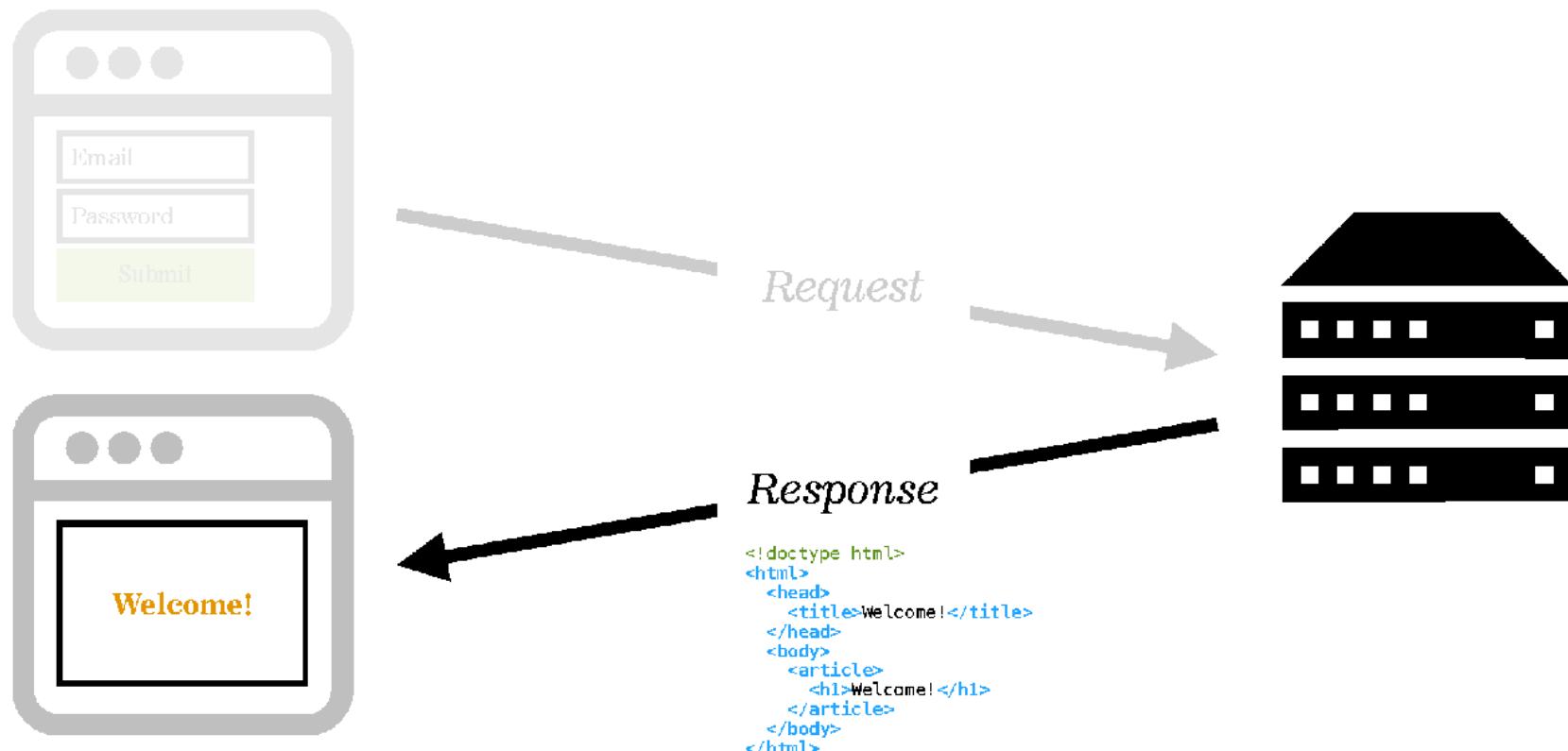
DAY 3 RECAP

91









REQUEST/RESPONSE DEVELOPER TOOLS

<http://login-form.bunts.io>

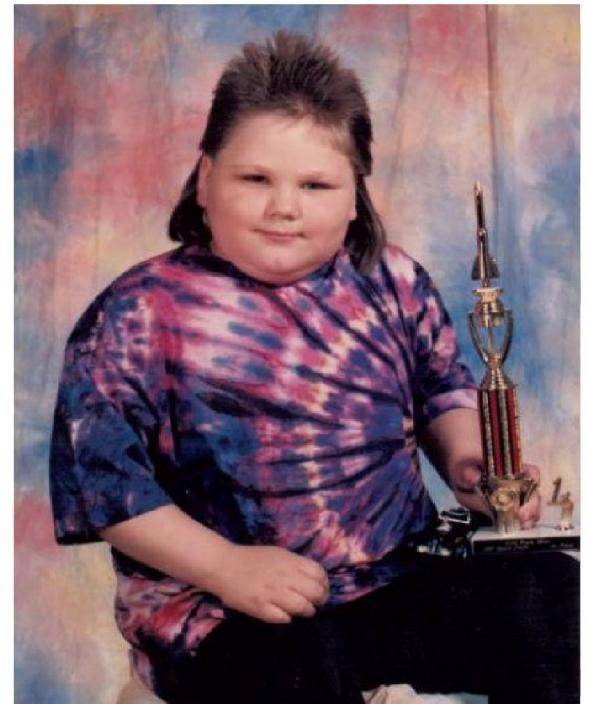
Source code:

<http://github.com/buntine/login-form>

APIS

Use APIs

- *Grab a picture from Reddit.com using their API*
- *Email that picture to a friend using the Mandrill API*

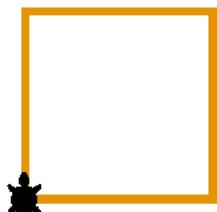


EXERCISE: TURTLE

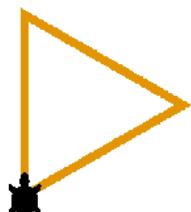
Imagine there was a turtle that understood the instructions (functions) on the right.

Break up into *groups of 2*

Write down a recipe (routine) for each of the pictures below, using the functions available.



Square



Triangle



Double Square

Documentation

Available turtle functions

- Forward x metres
- Turn right x degrees
- Turn left x degrees
- Repeat x times [
 - *instructions*]

ALGORITHMS

AKA FUNCTIONS, METHODS, ROUTINES



Square

```
Repeat 4 times [  
  Forward 5 metres  
  Turn right 90 degrees  
]
```



Triangle

```
Repeat 3 times [  
  Forward 5 metres  
  Turn right 120 degrees  
]
```



Double Square

```
Repeat 2 times [  
  Repeat 4 times [  
    Forward 5 metres  
    Turn right 90 degrees  
  ]  
  Turn right 45 degrees  
]
```

A CREATIVE PROCESS

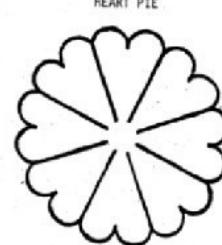
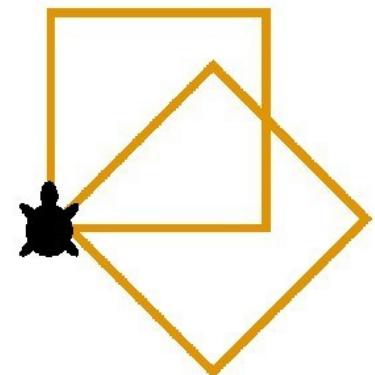
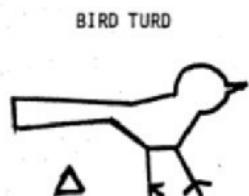
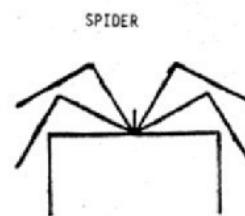
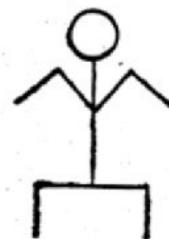
Understanding the problem

Exploring solutions

Pattern recognition

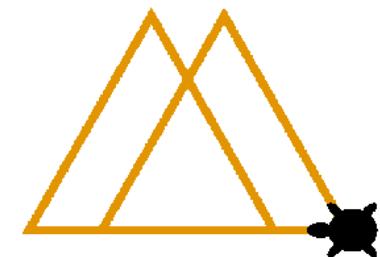
Abstraction

Sharing ideas



WHY ABSTRACT YOUR CODE?

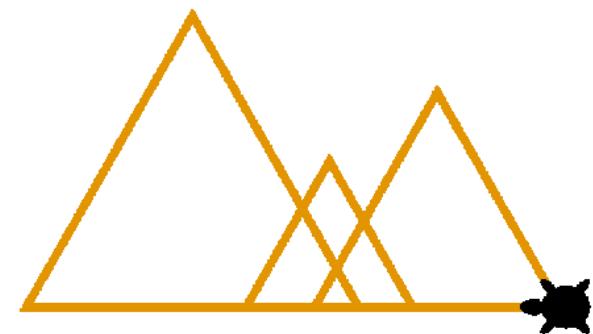
- Simplifies your code
- Allows yourself (and others) to re-use your work
- Libraries and frameworks provide pre-built abstractions



WHY USE PARAMETERS?

Benefits:

- Makes your code easier to read
- Makes your code more re-usable and generic



```
T0 triangle :size      left 90
repeat 3 [              triangle 80
    forward :size       forward 50
    right 120          triangle 50
]                         forward 20
END                        triangle 100
```

FIRST RUBY PROGRAM

- Open <https://nitrous.io/>
- Create a new Rails box
- Create a new file `hello.rb` inside Workspace with:
`puts "Hello world"`
- Save the file
- In the console:
`ruby hello.rb`

WHAT IS A WEB FRAMEWORK?

Toolkit for building web applications

Has library functions for handling:

- Requests
- Responses
- Templates
- Cookies

RAILS AND SINATRA

Rails has the concept of:

- Routes
- Controllers
- Views (templates)
- Models (database)
- Mailers (email)
- and more...

