
PROGRAMMING FOR NON-PROGRAMMERS

Antonio Lulic @antoniolulic

WIFI Network: GA-Guest

WIFI Password: yellowpencil

Install these for today (if you don't have them already):

Google Chrome <http://www.google.co.uk/chrome>

Sublime Text <http://www.sublimetext.com>

Ruby (for Windows) <http://rubyinstaller.org>

Day One Slides <http://bit.ly/gapfnp1>

Day Two Slides <http://bit.ly/gapfnp2>

PROGRAMMING FOR NON-PROGRAMMERS

Antonio Lulic

Day 2:

- Review Day 1 topics
- Beginning jQuery
- What is programming?
- Javascript
- What is back end development?
- Ruby
- What is an API?
- Upload your profile site to Git

WHAT IS JQUERY?

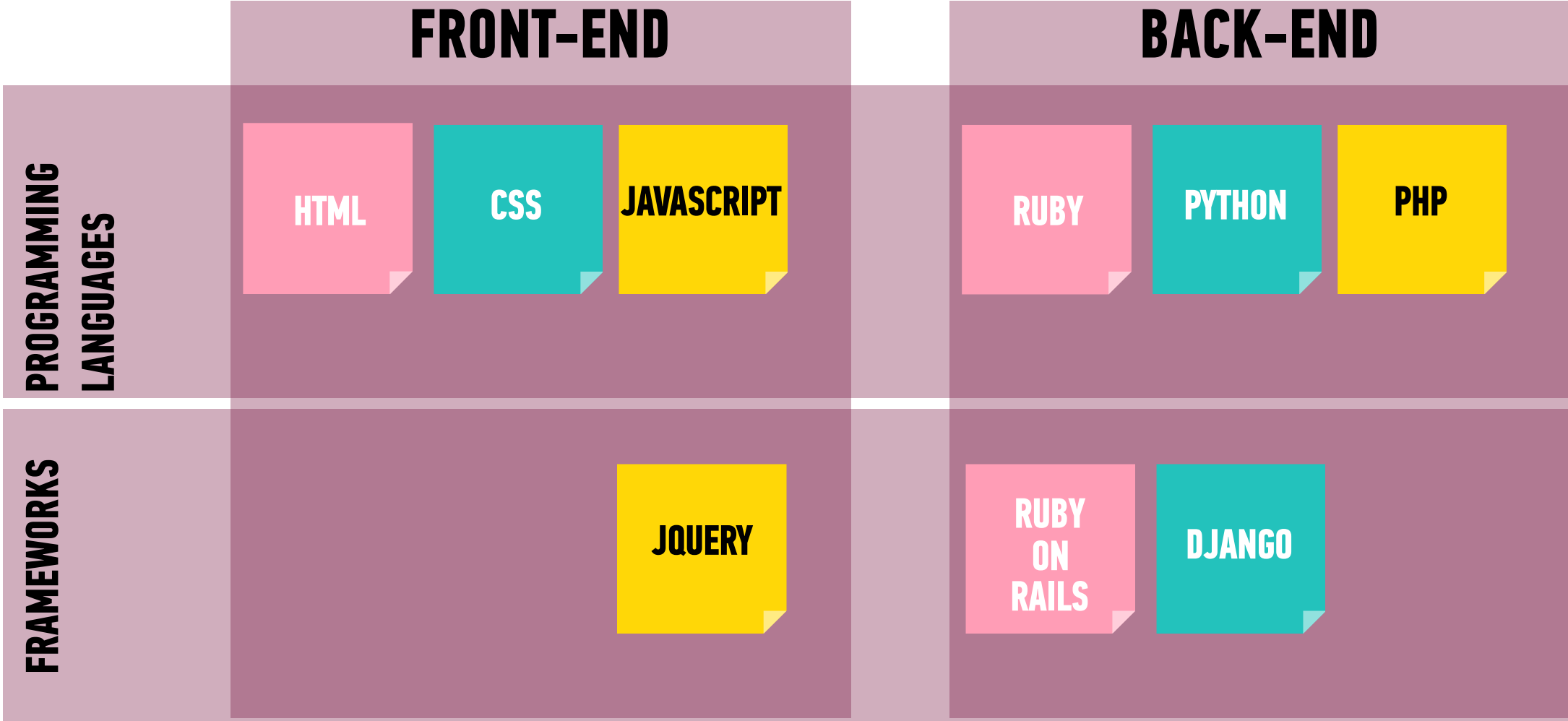
- We are going to learn a little jQuery today to make JS more fun!

HOW DO I DECIDE WHAT LANGUAGE TO USE?

WHAT TO LOOK FOR IN A LANGUAGE.

- Community support
- Difficulty level
- Development time
- Front-end or back-end?

FRONT END VS BACK END



INTRODUCTION TO JAVASCRIPT + JQUERY

WHAT IS JAVASCRIPT?

KEY WORDS

- Programming Language
- Behavior
- file extension = .js

PROBLEM SOLVING

JavaScript (really all programming languages) programs executes:

- linearly

 - One thing happens after another

- decisions

 - IF email notification received THEN download email

- storage

 - Archive the email I received

- iterations

 - Mark the next ten messages as unread

WHAT JAVASCRIPT CAN DO

- Examples!

ADDING JAVASCRIPT TO THE WEBPAGE

Loading a JavaScript file:

```
<script src="path/to/file.js"></script>
```

Inline JavaScript

```
<script>  
    "JavaScript code goes here";  
</script>
```

JAVASCRIPT STATEMENTS

A line of code.

Ends in a semicolon (;)

```
// Commented line
```

```
/*  
    Commented block - just like CSS!  
*/
```

BASIC JQUERY

ADDING JQUERY TO OUR WEBPAGES

```
<script src="http://code.jquery.com/jquery.js"></script>
```

DOCUMENT READY FUNCTION

```
$(function() {  
  
});
```

Key Topics:

- When is it executed

Notes:

At this point students will memorize the syntax.
We will discuss more in details later on.

JQUERY FUNCTIONS

```
<h1>General Assembly</h1>
```

```
<p>Hello</p>
```

```
$('p').html('goodbye');
```

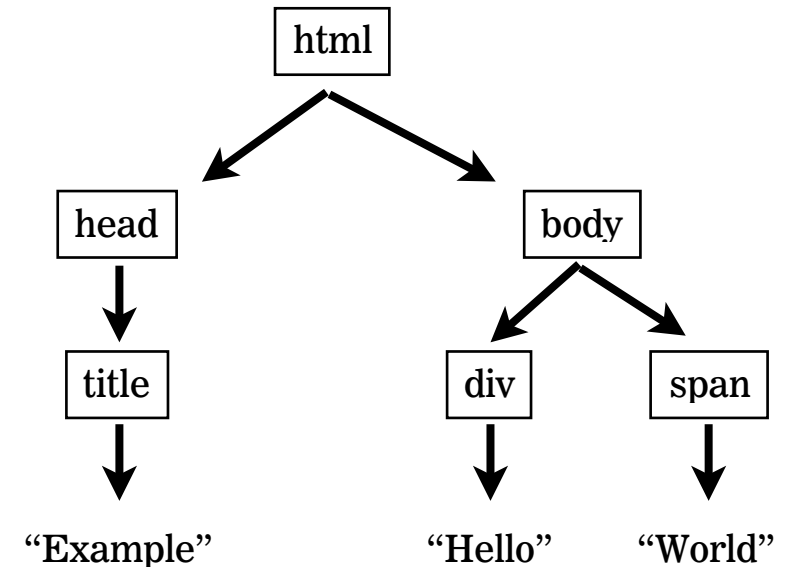
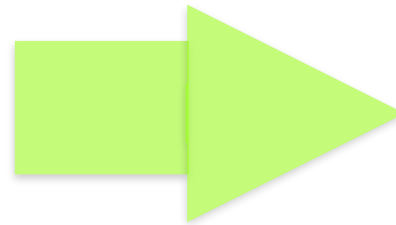
```
$('h1').css('color', 'blue');
```

You can use code pen to demonstrate.

DOCUMENT OBJECT MODEL (DOM)

The browser creates a model in its memory, like a family tree, of the elements that your HTML describes.

```
<html>  
  <head>  
    <title>Example</title>  
  </head>  
  <body>  
    <div>Hello</div>  
    <span>world</span>  
  </body>  
</html>
```



HTML VS DOM

- The browser is showing you the HTML + CSS that is stored in the DOM.
- If we change things in the DOM, that changes are shown to us instantly!
- Things like
 - new/removed elements
 - new/removed attributes
 - new/removed styles

FUNDAMENTALS

- \$ function
- Finding elements on a page, use selectors with the \$ function:
 - `$("#searchBox")`
 - `$(".errors")`
 - `$("p")`
- Returns an array of HTML elements you have selected.

MANIPULATING STYLE

```
.css('property', 'value');
```

Changes the CSS values for all matched elements.

```
.css({'property': 'value', 'property': 'value'});
```

Change multiple CSS property values at once.

```
.css('property');
```

Similar to `.html()`, when second arguments is not given, it returns the current value of the CSS property in question.

MANIPULATING HTML

.html()

With no argument, it's returns the html inside the matched element

.html('<p>some html</p>');

Inserts (and overwrites!) the html inside the selected elements with the htmlString

.text()

Can be used in both XML and HTML documents. The result of the .text() method is a string containing the combined text of all matched elements.

.text('Some text')

Replaced the content of the matched elements with the provided text.

REVIEW

- jQuery (jquery.com)
 - ›a JavaScript library that makes DOM manipulation simple.
 - ›“Cross browser”
 - ›works the same in all* browsers.
 - ›allows:
 - ›document traversal
 - ›css manipulation
 - ›event handling
 - ›animation

THE CONSOLE

WHERE IS THE JAVASCRIPT CONSOLE?

Why would you use this?

OUTPUT LOG MESSAGES TO THE CONSOLE

In the Developer Tools console type:

- `console.log('Hello world');`
- Your output goes between the parentheses

INPUT FROM USERS

```
prompt('What is your first name?');
```

VARIABLES & DATA

The slides are used to help students understand what is a data type.

WHAT IS A VARIABLE?

VARIABLES

Declaration:

- ▶ `var age;`

Assignment:

- ▶ `age = 21;`

Declaration and initialization

- ▶ `var age = 21;`

RE-ASSIGNMENT

```
var name = "Jo";
```

```
name = "Mich";
```

WHAT CAN VARIABLES STORE?

DATA TYPES

We will look at:

- string
- number
- boolean

STRINGS

Stores textual information

- Double quotes
 - "How is the weather today?"
- Single Quotes
 - 'Warm'

QUOTES IN STRINGS

Double vs single quoted strings:

- ▶ 'They "purchased" it'
- ▶ "It's a beautiful day"

Escaping

"They've \"purchased\" it"
'It\'s a beautiful day'

NUMBERS

Represent numerical data

►int: 42

►float: 3.14159265

Signed

►int: +6

►float: -8.2

Can perform arithmetic on number data types

ARITHMETIC IN JAVASCRIPT

Operator	Description	Example
+	Addition	1+1
-	Subtraction	3-2
*	Multiplication	5 * 3
/	Division	10 /2
++	Increment	5 ++
--	Decrement	5 --
%	Modulus	1%2

BOOLEAN

Binary, two possible values:

- ▶ true
- ▶ false

Has driver license:

- ▶ If driver has license: true
- ▶ If driver does not have license: false

VARIABLE CONVENTIONS

Variables start with a lower case letter

If they contain multiple words, subsequent words start with an upper case letter

▶e.g: `var numberOfStudents = 10;`

DEBUG VARIABLES – UNDEFINED

```
var name = "Jo";
```

```
name.surname;
```

‣ surname property is not on name, therefore it's undefined

DEBUG VARIABLES – NULL

- ▶ `var colour = null;`
- ▶ `var size; //This is null and undefined`

DEBUG VARIABLES – NULL VS UNDEFINED

`var amount;`

`amount is null`

`amount is also undefined`

A FEW BASIC OPERATIONS

Length of a string:

- ▶ `var name = "Jo";`

- ▶ `name.length`

- ▶ 2

- ▶ Can be done directly on the string:

- `"Jo".length`

- ▶ 2

DATA TYPE CONVERSION

When/ why would you convert a data types?

CONVERSION: STRING TO NUMBER

```
var age = "4";
```

```
var convertedAge = parseInt(age);
```

```
var pi = "3.14159";
```

```
var convertedPi = parseFloat(pi);
```

- These work:
 `parseInt("4");`
 `parseFloat("3.14159");`
 `parseInt("3.5");` //gives 3

CONVERT: NUMBER TO STRING

```
var number = 4;  
    ▶number.toString();    "4"
```

OR

```
number + "";    "4"
```

PROBLEM CONVERSION: NUMBER TO STRING

This results in an error:

- `4.toString();`

This does not:

- `(4).toString();`

Why?

PROBLEM CONVERSION: STRING TO NUMBER

```
var notANumber = parseInt("blahblah");
```

```
var nullThing = null;
```

```
parseFloat(nullThing);
```

CONDITIONAL LOGIC

COMPARISONS

Why would you need to compare.

Relate to robot activity
or something concrete.

People seem to relate to
the password verification

COMPARISONS – EQUALITY

- Are two things equal?

- `10 === 10`

- `true`

- `10 === 5`
`false`

- `“hi” === “hi”`

- `true`

$$x = 3$$

Logical Operators			
Operator	Description	Comparing	Returns
==	equal to	$x == 8$	FALSE
===	exactly equal to(value and type)	$x === "3"$	FALSE
		$x === 3$	TRUE
!=	is not equal	$x != 8$	TRUE
!==	is not equal(neither value nor type)	$x !== "3"$	TRUE
		$x !== 3$	FALSE
>	greater than	$x > 8$	FALSE
<	less than	$x < 8$	TRUE
>=	greater than or equal to	$x >= 8$	FALSE
<=	less than or equal to	$x <= 8$	TRUE

CONDITIONALS

- What is a Conditional?
- Why would we use it? (Remember your robot example)

THE SYNTAX - IF

```
var topic = "JS";
```

```
if (topic == "JS") {  
    console.log("You're learning JavaScript");  
}
```

CONDITIONALS - IF ELSE

```
if (condition is true) {  
  
    console.log("The condition is true");  
  
} else {  
  
    console.log ("The condition was false");  
  
}
```

IF/ELSE-IF/ELSE

```
var topic = "JS";

if (topic == "JS") {
    console.log("You're learning JavaScript");
} else if (topic == "JavaScript") {
    console.log("You're still learning JavaScript");
} else {
    console.log("You're learning something else");
}
```

COMPARING MULTIPLE CONDITIONS: &&

```
if (name == "GA" && password == "YellowPencil") {  
    console.log ("You can access the internet");  
}
```


COMPARING MULTIPLE CONDITIONS: ||

```
if (day == "Tuesday" || day == "Thursday"){  
    console.log ("You have class today")  
}
```

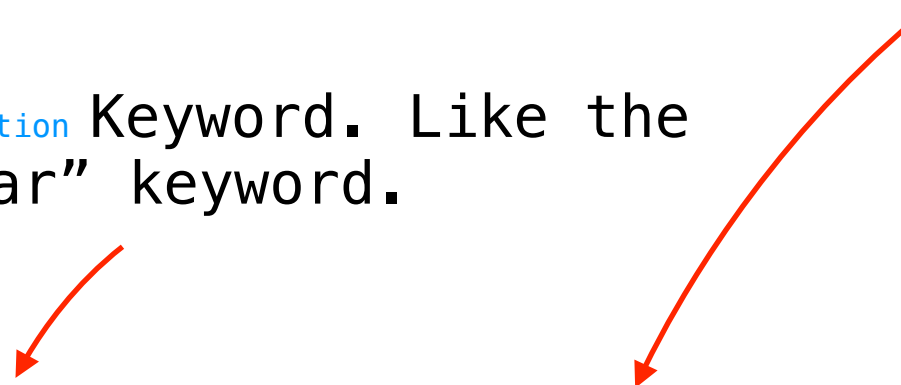
FUNCTIONS

FUNCTIONS - SYNTAX

`function` Keyword. Like the
“var” keyword.

The name of your function


```
function functionName(arg1, arg2) {  
    //Body of function  
}
```

A diagram with two red curved arrows. One arrow starts from the text 'function Keyword. Like the “var” keyword.’ and points to the word 'function' in the code snippet. The other arrow starts from the text 'The name of your function' and points to the word 'functionName' in the code snippet.

FUNCTIONS – SYNTAX

Arguments let you pass data into the function

```
function functionName(arg1, arg2) {  
    //Body of function  
}
```

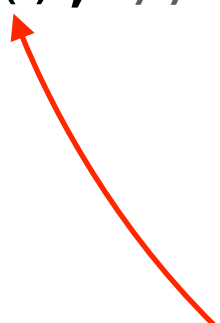
A diagram with two red arrows. One arrow starts above the text 'Arguments let you pass data into the function' and points down to the arguments 'arg1, arg2' in the function signature. The other arrow starts below the text 'The functions executed code goes between the { } brackets. Much like an “if” statement.' and points up to the body of the function, which is the code between the curly braces.

The functions executed code goes between the { } brackets. Much like an “if” statement.

FUNCTIONS – EXAMPLE

```
function helloWorld() {  
  console.log("Hello Functions");  
}
```

```
helloWorld(); //Prints "Hello Functions to the  
console.
```



The brackets execute the function. Try calling the function without them to see what happens.

FUNCTIONS - EXAMPLE

```
function addAndPrint(num1, num2) {  
    var sum = num1 + num2;  
    console.log(sum);  
}
```

```
addAndPrint(1, 2); // Result is 3
```

```
addAndPrint(8, 2); // Result is 10
```

RETURNING DATA FROM FUNCTIONS

- What if we want to use the data the function creates?
- The “return” method allows us to do that.

FUNCTIONS - EXAMPLE

```
function add(num1, num2) {  
    var sum = num1 + num2;  
    return sum;  
}
```

```
add(1, 2); // 3 is the result
```

```
var answer = add(20, 10);
```


ORGANIZING FUNCTIONS

- How will you explain where functions go?
- How will you describe where they should go when you are using a `document.ready()` functions?

JAVASCRIPT OBJECTS

OBJECTS – FORMAL DEFINITION

- An “object” in computer science is a collection of data and functions that work with that data.
- Objects allow us to organize similar data effectively

EXAMPLE OF AN “OBJECT”

- Person
 - Has a name, age, location
 - Can speak, eat.
- Lightbulb
 - Number of watts, brand
 - Can turn on and off

JAVASCRIPT OBJECTS - SYNTAX

Empty object {}

Person Object

```
var person = {  
  age: 20,  
  name: "Kevin Bacon"  
};
```

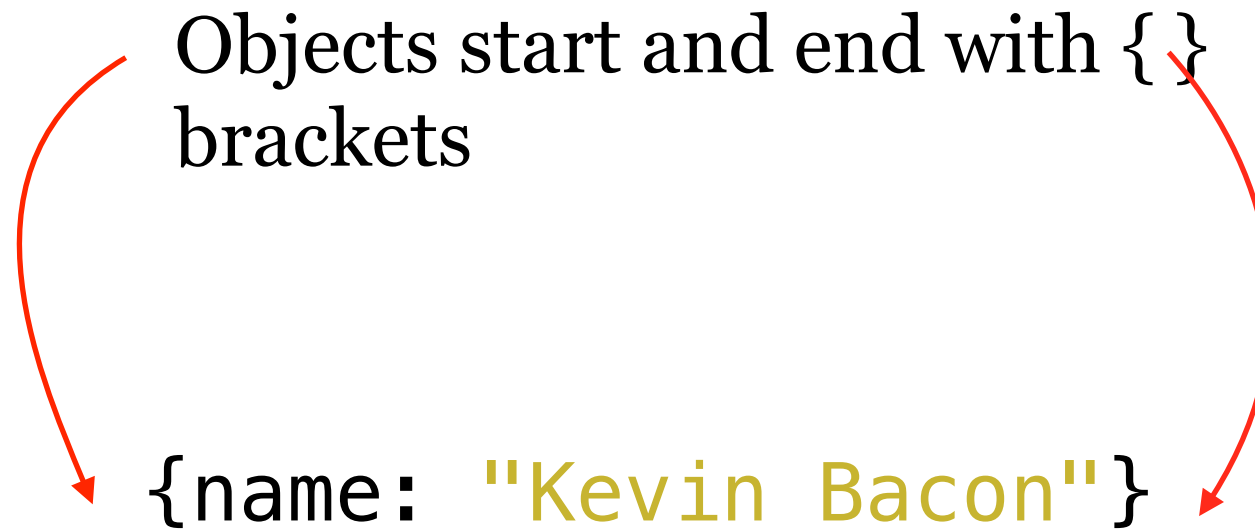
SYNTAX OF THE “KEY: VALUE” PAIRS

```
{key: "Value"}
```

SYNTAX OF THE “KEY: VALUE” PAIRS

```
{  
  age: 20,  
  name: "Kevin Bacon",  
  profession: "Actor"  
};
```

SYNTAX OF THE “KEY: VALUE” PAIRS



SYNTAX OF THE “KEY: VALUE” PAIRS

The key is similar to a variable name.



```
{name: "Beyonce Knowles"}
```

SYNTAX OF THE “KEY: VALUE” PAIRS

```
{name: "Beyonce Knowles"}
```


The value can be anything!
String, number, boolean..
even function or another
object!



SYNTAX OF THE “KEY: VALUE” PAIRS

```
{name: "Beyonce Knowles", age:  
20};
```

Multiple key/value pairs are separated by commas, like array values.



INDEX METHOD - ACCESSING DATA

Creating

```
var test = {a: "hi"};
```

Accessing

```
test["a"];  
// returns "hi"
```

Assigning

```
test["a"] = "bye";  
// test["a"] now  
stores "bye"
```

DOT METHOD – ACCESSING OBJECTS

Creating

```
var test = {a: "hi"};
```

Accessing

```
test.a;  
// returns "hi"
```

Assigning

```
test.a = "bye";  
// test["a"] now  
stores "bye"
```

CAR EXAMPLE

```
var car = {  
  make: "Ford",  
  model: "Focus",  
  year: 2013,  
  mileage: 89000  
}
```

APIS

WHAT IS AN API?

Application Programming Interface

SHOULD I USE AN API?

Wait, what's an API?



EVERNOTE Developers



Developers

facebook developers



DEVELOPERS

API DEMO

DATABASES

WHAT IS A DATABASE?

An organised collection of information.

- MySQL
- PostgreSQL
- Oracle
- MongoDB



SQL

Structured Query Language

```
SELECT * from Books WHERE price > $10
```

ID	TITLE	AUTHOR	PRICE
1	I Know Why the Caged Bird Sings	Maya Angelou	\$12
2	I Sing the Body Electric	Ray Bradbury	\$14
<u>3</u>	Of Mice and Men	John Steinbeck	\$11
<u>4</u>	The Proper Study	Isaac Asimov	\$11
<u>5</u>	Such, Such Were the Joys	George Orwell	\$11.5
<u>6</u>	The Waste Land	T. S. Eliot	12.5
7	The Golden Apples of the Sun	Ray Bradbury	\$13.99

CRUD

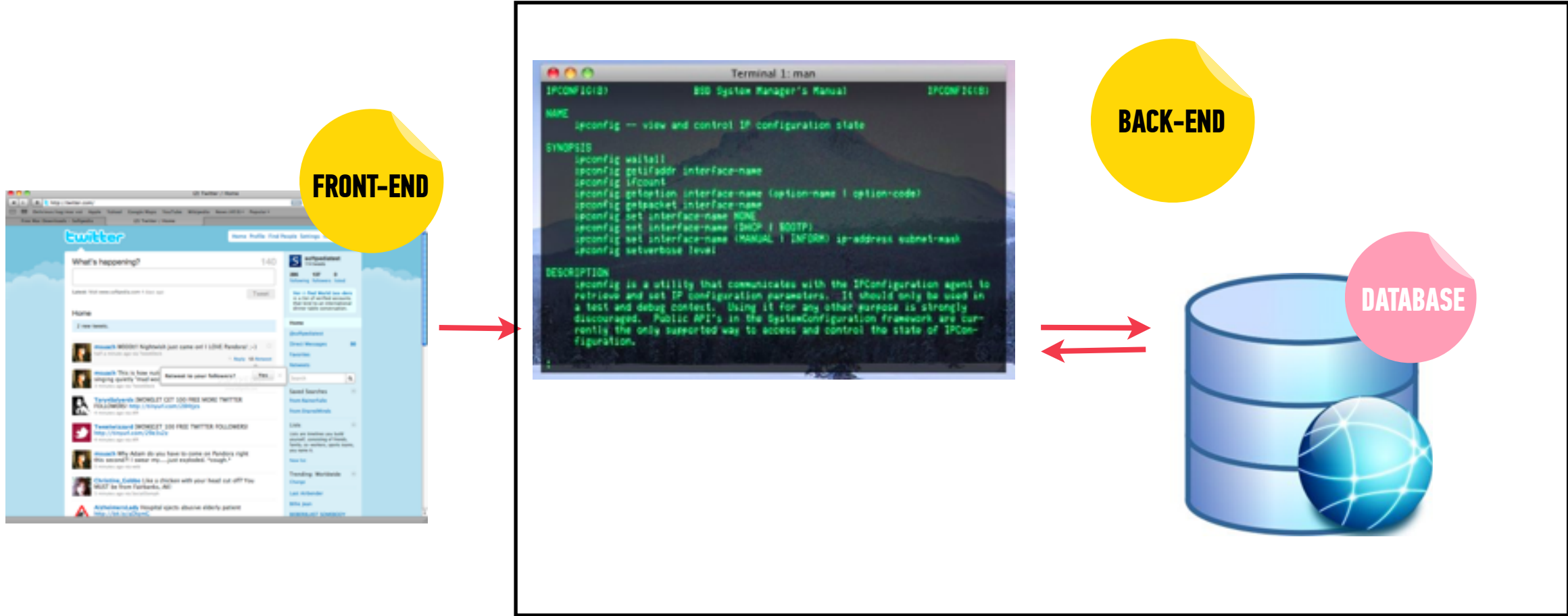
CREATE

READ

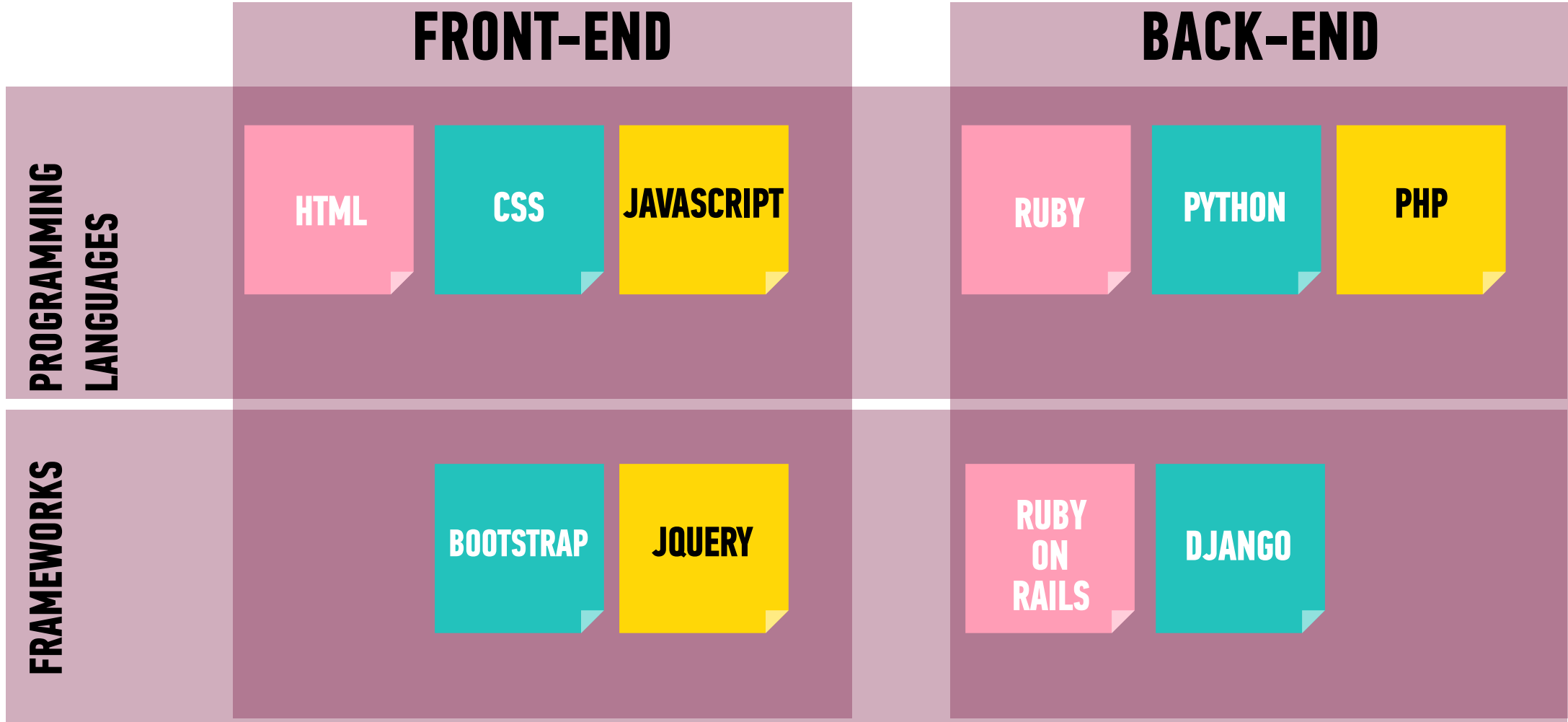
UPDATE

DESTROY

WHAT IS BACK-END WEB DEVELOPMENT?



FRONT-END VS. BACK-END



RUBY

SINATRA

RUBY ON RAILS

PROGRAMMING CONCEPTS (USING RUBY)

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."

GETTING SETUP

Is Ruby downloaded and installed?

```
ruby -v
```

```
ruby 1.9.3p194    <-- you should get  
                    something like this
```



VARIABLES

VARIABLES

STORING AND DISPLAYING THINGS

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

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

DATA TYPES

BASIC DATA TYPES

STRING e.g. "Hello"

NUMBER e.g. 58

BOOLEAN e.g. true/false

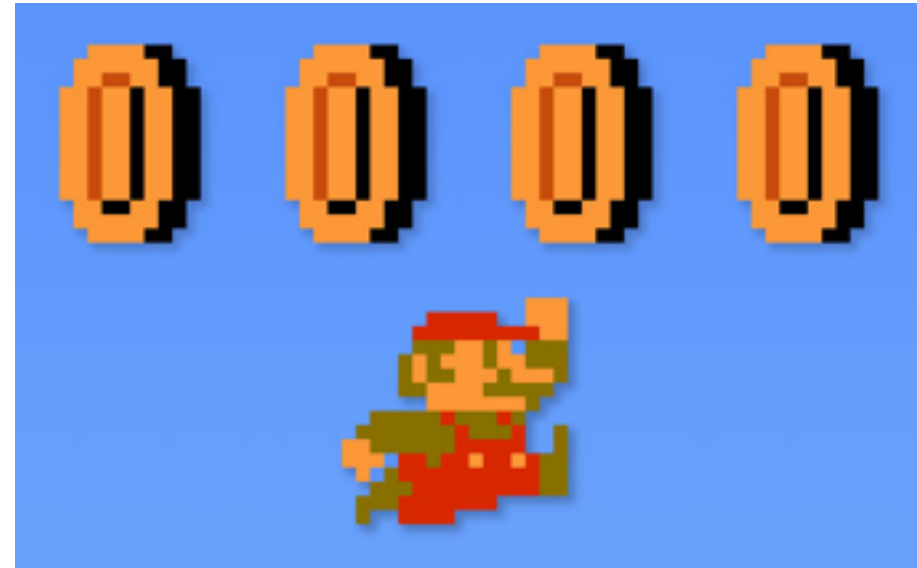
CONDITIONALS

CONDITIONALS

if age > 18 puts “you are an adult”

if thing == “coin” then score = score + 1

score: 4



COMPARISON OPERATORS

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

LOOPS

LOOPS

REPEATING THINGS

#print your name 10 times

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

FUNCTIONS

FUNCTIONS

e.g.

```
def say_hello_to(name)
  puts "Hi #{name}"
end
```

```
say_hello_to "John"
```

LET'S PROGRAM IN THE RUBY

VERSION CONTROL

- Keep track of the version of code.
- Collaborate with others.
- Keep track of who contributed.
- Open-source (release it to the world for free!)
- Popular tools: GitHub, BitBucket



LEARNING RESOURCES

GA Dash - dash.generalassembly.ly
Codecademy
Code School
Tuts+
Ruby docs
NetTuts