BEWD - Variables and Conditional Logic

Matt Heath
Tech Lead, Platform
Hailo

AGENDA

- Intro to Ruby & irb
- Variables
  - Numbers
  - Strings
  - Booleans
- Method Basics
- Conditional Logic
- Lab Time

RUBY

A PROGRAMMING LANGUAGE

- An open source programming language
- Easy to read and natural to write
- Created by Yukihiro Matsumoto (*aka Matz*) with the goal of building a language for developers
- Regularly maintained and evolved (recently reached version 2.1.2)

RAILS

A WEB APPLICATION FRAMEWORK

- Open source web application framework built in Ruby
- Allows you to create web applications that query a database
- Created by DHH (David Heinemer Hansson) to simplify the task of building web applications, using *conventions*

RUBY & RAILS

RUBY FIRST

- It will be easier to navigate a Rails project once we have a basic understanding of Ruby
- We will first teach you how to write simple Ruby scripts as stand-alone applications
- Once we have become familiarised with Ruby, we will start building Rails applications (which are essentially groups of Ruby script files that work together)

COMPUTATIONAL THINKING

WHAT DOES IT MEAN TO PROGRAM?

"Learning about "for" loops is not learning to program, any more than learning about pencils is learning to draw."

– Bret Victor, Learnable Programming


LAB

ROBOT RECAP

CHANGING HOW YOU THINK

- Think in logical steps to solve a problem
- Use Ruby keywords to help solve those problems
  - Conditional Logic
  - Iteration

ROBOT RECAP

PSEUDO CODE

- Connect to server
- Get the number of unread emails
- If the number of unread emails is zero, exit
- For each of the unread emails read the subject and the body of the email, then mark as read

FUNDAMENTALS

- In order to start writing our own Ruby programs, we need to learn some of the basic fundamental programming tools

- Specifically, we need to learn:
  - Variables
  - Methods
  - Conditions

SAVING VALUES

USING VARIABLES

- We can tell our program to remember values for us to use later on
- The action of saving a value to memory is called **assignment**
- The entity we use to store the value is called a **variable**
- The action of getting the value from a variable is called **accessing** the variable
- We will use all the above techniques to store values into variables, and generate new values using existing variables

VARIABLES

STORING VALUES

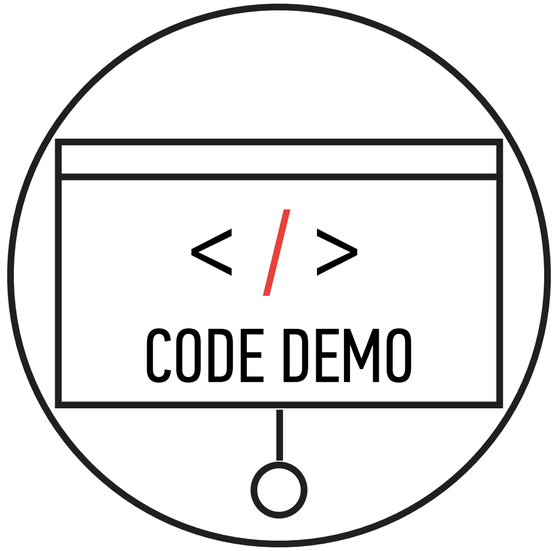
```
>> name = "Steven"
 => “Steven"

>> age = 2013 - 1983
 => age # 30
```

DATA TYPES

- The types of different values we support include numbers, text, and other more complex ones we'll see in the future

- Ruby has its own names for these:

```
1                          #Fixnum
1.99                       #Float
'Hi! String here!'         #String
"I'm a string too!"        #String
```


< / >
CODE DEMO

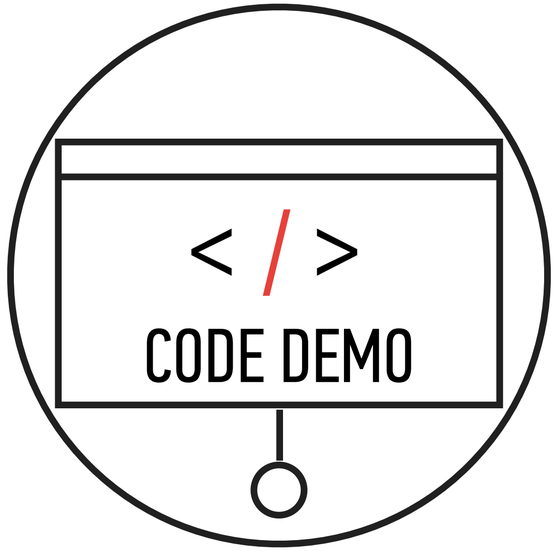
SAVING CODE

USING METHODS

- The same way we can store VALUES in memory by using variables…
- We can store CODE in memory by using methods.
- In other words, we can train the program to 'remember' a set of commands, and give that set of tasks a command name
- Then, we can call that command by name and the program will perform those tasks

ARITHMETIC OPERATORS

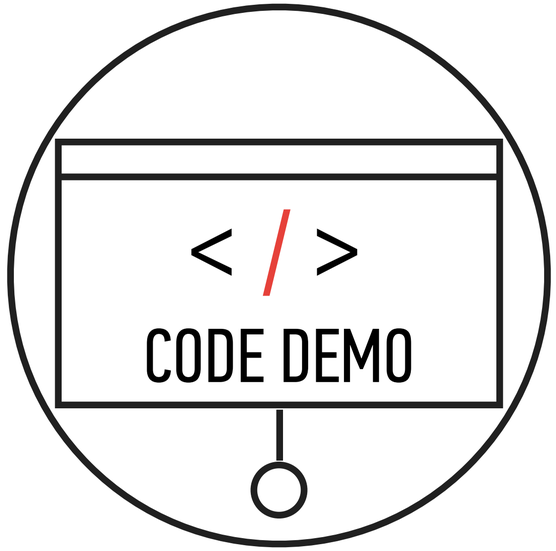
Operator	Meaning	Example
+	Addition	8 + 10
-	Subtraction	10 – 8
*	Multiplication	12 * 2
/	Division	10 / 5
%	Modulus	10 % 6


< / >
CODE DEMO

METHODS

RECAP

- Methods let us train the program to 'remember' a set of code to perform later
- Making a new method is called **declaring** a method
- Declaring a method does NOT run the method immediately
- If the method takes in variables to use while it is doing its tasks, those are called **parameters**


< / >
CODE DEMO

DATA TYPES

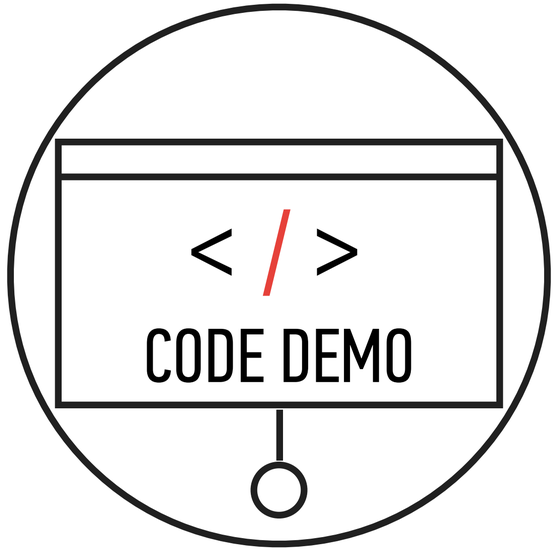
BOOLEANS

- In addition to strings and integers, Ruby also has a Boolean data type
- A boolean is a simple value that is either '**true**' or '**false**'
- When different data types are compared to each other, the result of that comparison is a boolean result

```
5 < 7
=> true
```

LOGIC OPERATORS

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


< / >
CODE DEMO

VARIABLES & DATA TYPES

RECAP

Data Types

- Number
- Float (number with decimals)
- String
- Booleans

Variables

- Store values
- Can be passed to methods as parameters

CONDITIONAL LOGIC

MAKING DECISIONS

It's either TRUE or FALSE (like booleans)

If you are greater than 18
you are an adult

```
if age > 18
  puts "You are an adult"
end
```

CONDITIONAL LOGIC

MULTIPLE CONDITIONS

```
guess = 7
if guess > 5
  puts "Too high!"
elsif guess < 5
  puts "Too Low!"
else
  puts "You've guessed my digit!"
end
```

CONDITIONAL LOGIC

MULTIPLE CONDITIONS

Operator	Description	Example
and	Called Logical AND operator. If both the operands are true then then condition becomes true.	(A and B) is true.
or	Called Logical OR Operator. If any of the two operands are non zero then then condition becomes true.	(A or B) is true.
&&	Called Logical AND operator. If both the operands are non zero then then condition becomes true.	(A && B) is true.
||	Called Logical OR Operator. If any of the two operands are non zero then then condition becomes true.	(A || B) is true.
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(A && B) is false.


LAB