T1A3 - Terminal Application

TRIVIA TIME

Mike Olivotto

Contents

- Terminal app walk through
  - Features & how it’s used
  - Logic of the code
- Development & Build Process
  - Challenges
  - Ethical Issues
  - Favourite Parts
- Demo

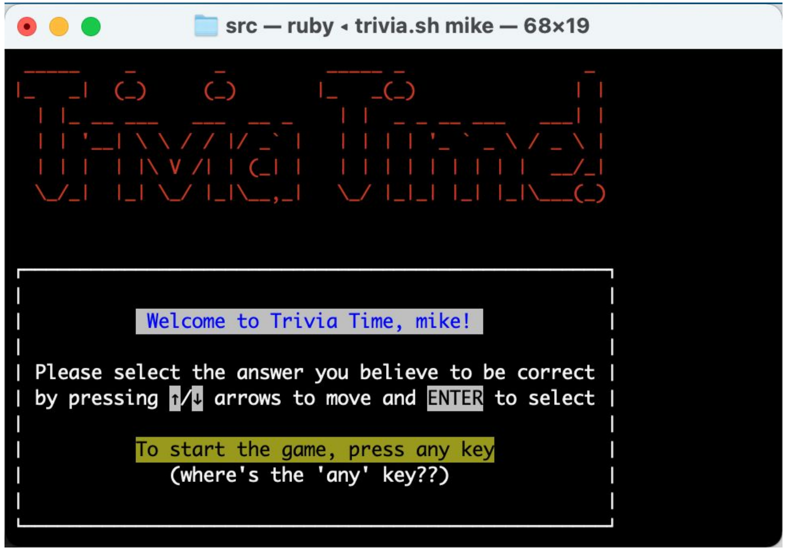
App walkthrough - Features / Structure

Features

- 3 difficulty modes (easy, regular, hard)
- Questions w/ multi-choice answers
- Display score
- Display correct answers

Structure

- index.rb
  - Handles setup arguments (name, difficulty)
  - Instantiates the game
- Trivia_game.rb
  - Contains TriviaGame class
- JSON files
  - Containing Qs + As. One file per difficulty mode
- Gemfiles, bash script, RSPEC files...



src — ruby ◂ trivia.sh mike — 68×19

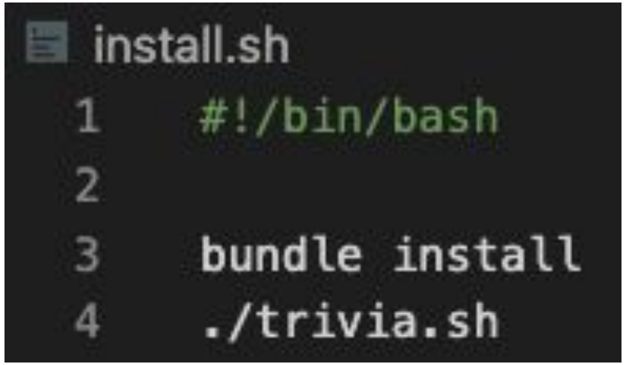
```
 _____     _       _         _____ _                _ 
|_   _| __(_)_   _(_) __ _  |_   _(_)_ __ ___   ___| |
  | || '__| \ \ / / |/ _` |   | | | | '_ ` _ \ / _ \ |
  | || |  | |\ V /| | (_| |   | | | | | | | | |  __/_|
  |_||_|  |_| \_/ |_|\__,_|   |_| |_|_| |_| |_|\___(_)


 ----------------------------------------------------------
|                                                          |
|           Welcome to Trivia Time, mike!                  |
|                                                          |
| Please select the answer you believe to be correct       |
| by pressing ↑/↓ arrows to move and ENTER to select       |
|                                                          |
|           To start the game, press any key               |
|              (where's the 'any' key??)                   |
|                                                          |
 ----------------------------------------------------------
```

App walkthrough - Installation

Ruby is the minimum requirement for running this application. It is also recommended that you also have the Bundler gem installed on your machine.

- Install the app from the ‘/src’ directory with the installation file. Type ‘./install.sh’

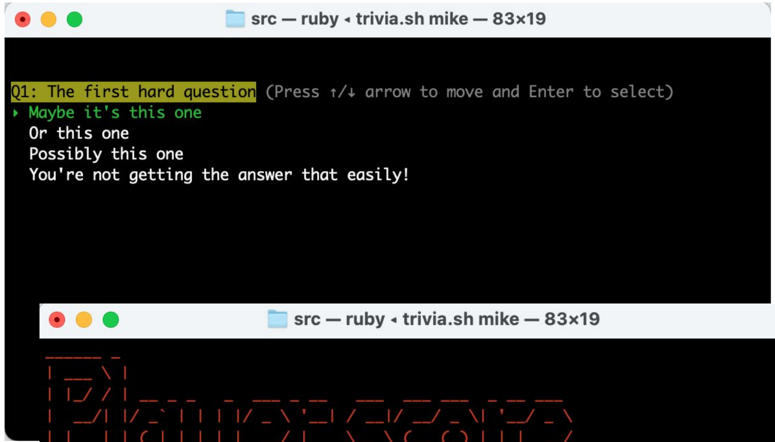
This file will run Bundler to install the dependencies and run the app for the first time


install.sh
```
#!/bin/bash

bundle install
./trivia.sh
```

App walkthrough - How it's used

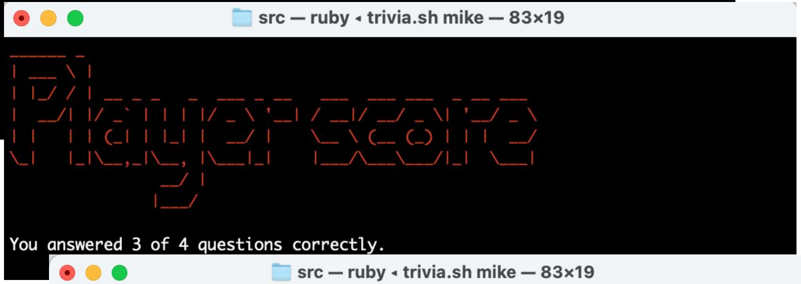
How it's used

- Load up:
  - ./trivia.sh [name] [mode] [-h/--help]
  - ruby index.rb [name] [mode] [-h/--help]
  - Arguments are not required for execution
    - Program will request name + difficulty if not entered
- User is presented a welcome screen + instructions. Press any key to proceed.
- User is presented with a question, uses ↑/↓ keys to select answer
- Upon completion, again with ↑/↓ keys, user can choose to:
  - View their score
  - Check the correct answers
  - Exit the app



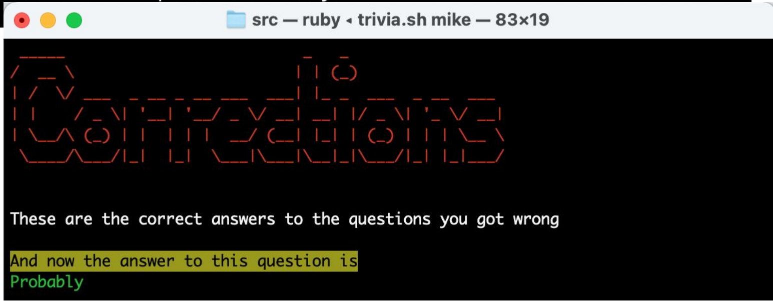
src — ruby ◂ trivia.sh mike — 83×19

```
Q1: The first hard question (Press ↑/↓ arrow to move and Enter to select)
‣ Maybe it's this one
  Or this one
  Possibly this one
  You're not getting the answer that easily!
```



src — ruby ◂ trivia.sh mike — 83×19

```
You answered 3 of 4 questions correctly.
```



src — ruby ◂ trivia.sh mike — 83×19

```
These are the correct answers to the questions you got wrong

And now the answer to this question is
Probably
```

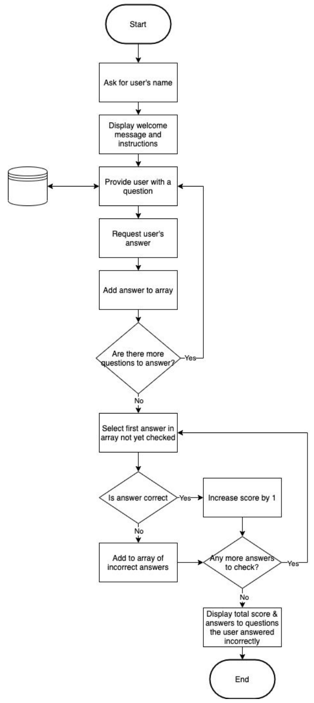
App walkthrough - Logic of the code

The basics

- Pull questions from a JSON file
- Serve each question, one at a time, with multiple-choice answers
  - Save user's answer to an array
- After all questions are exhausted, check user's answers against correct answers in the question file
  - Count up number of correct answers
- Display user score
- Display correct answers to to questions user got wrong

Multi-choice question/answer app

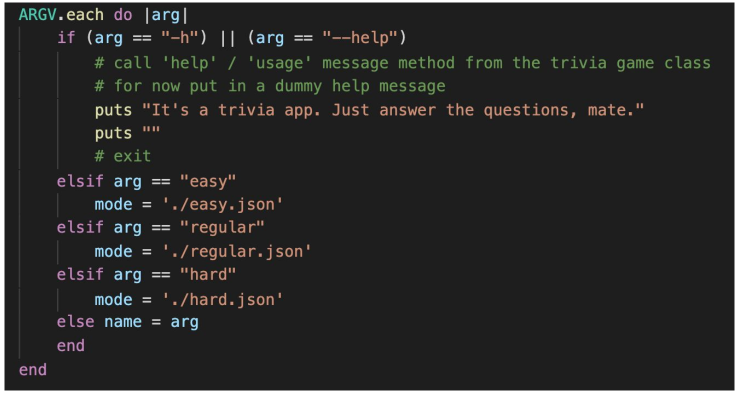
A terminal app written in Ruby, designed to serve the user with multiple-choice questions, providing them with an overall score and the correct answers to questions they answered incorrectly.


Start
Ask for user's name
Display welcome message and instructions
Provide user with a question
Request user's answer
Add answer to array
Are there more questions to answer?
Yes
No
Select first answer in array not yet checked
Is answer correct
Yes
Increase score by 1
No
Add to array of incorrect answers
Any more answers to check?
Yes
No
Display total score & answers to questions the user answered incorrectly
End

App walkthrough - Logic of the code

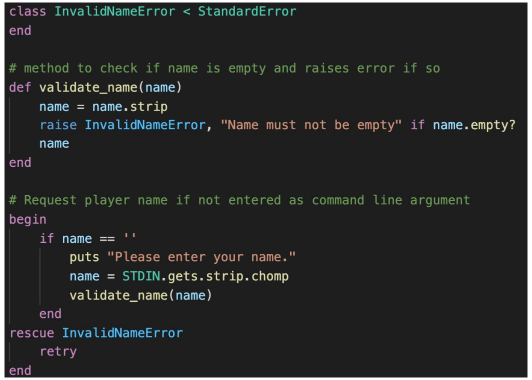
Finer details

- Accept command line arguments to set up name and mode
  - ‘if’ logic to allow for arguments in any order



```ruby
ARGV.each do |arg|
    if (arg == "-h") || (arg == "--help")
        # call 'help' / 'usage' message method from the trivia game class
        # for now put in a dummy help message
        puts "It's a trivia app. Just answer the questions, mate."
        puts ""
        # exit
    elsif arg == "easy"
        mode = './easy.json'
    elsif arg == "regular"
        mode = './regular.json'
    elsif arg == "hard"
        mode = './hard.json'
    else name = arg
    end
end
```

- Error handling
  - Custom exception when name is left empty
  - TTY-Prompt to handle all other input

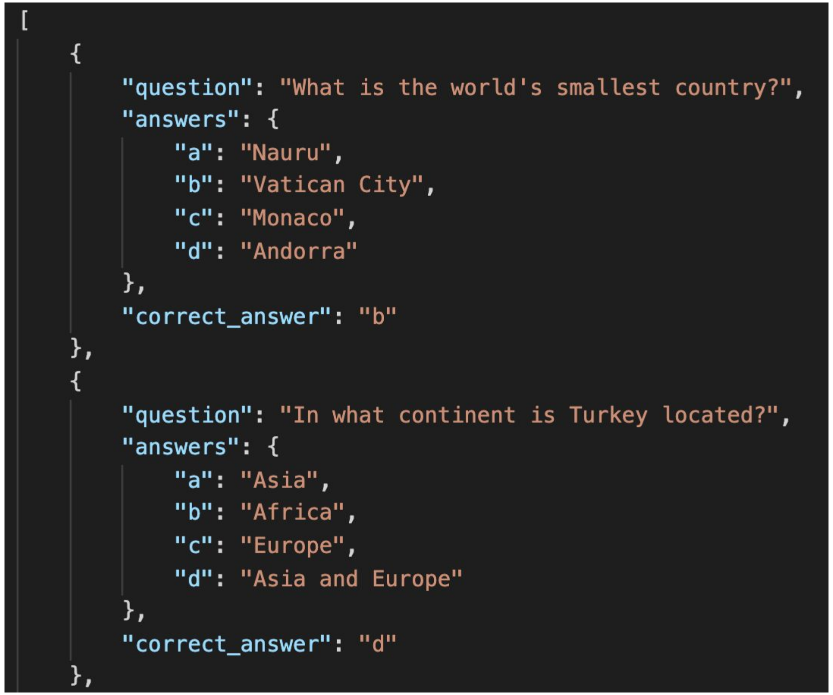


```ruby
class InvalidNameError < StandardError
end

# method to check if name is empty and raises error if so
def validate_name(name)
    name = name.strip
    raise InvalidNameError, "Name must not be empty" if name.empty?
    name
end

# Request player name if not entered as command line argument
begin
    if name == ''
        puts "Please enter your name."
        name = STDIN.gets.strip.chomp
        validate_name(name)
    end
rescue InvalidNameError
    retry
end
```

App walkthrough - Logic of the code

Finer details

- JSON files
    - Arrays containing objects (key/value pairs)

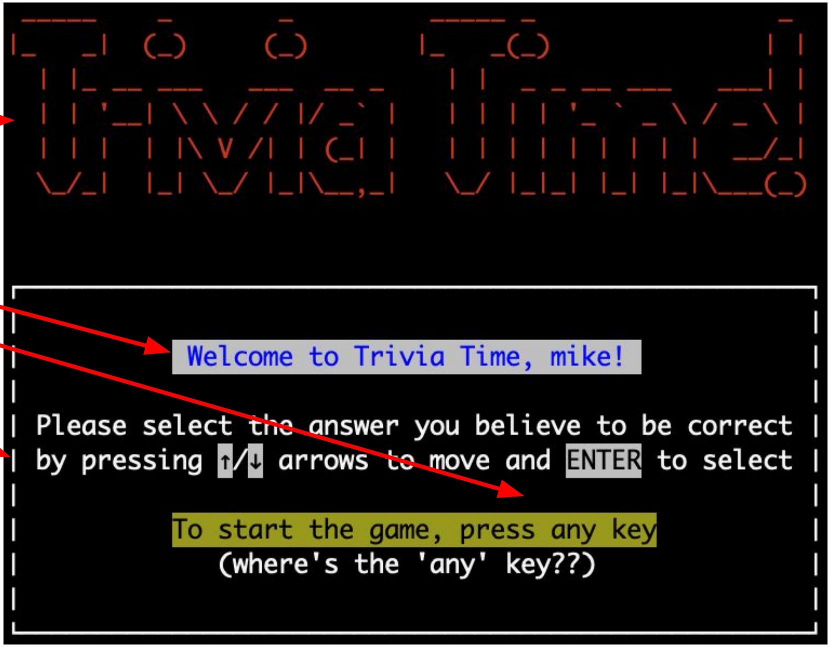


```
[
    {
        "question": "What is the world's smallest country?",
        "answers": {
            "a": "Nauru",
            "b": "Vatican City",
            "c": "Monaco",
            "d": "Andorra"
        },
        "correct_answer": "b"
    },
    {
        "question": "In what continent is Turkey located?",
        "answers": {
            "a": "Asia",
            "b": "Africa",
            "c": "Europe",
            "d": "Asia and Europe"
        },
        "correct_answer": "d"
    },
```

App walkthrough - Logic of the code

Finer details

- Gems
  - Artii
  - Colorize
  - TTY-prompt
  - TTY-box

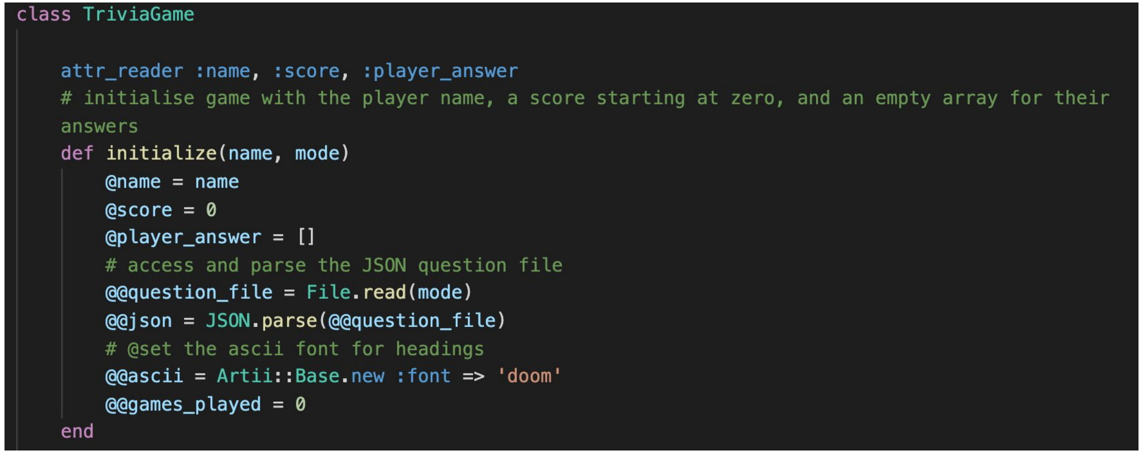


```
Welcome to Trivia Time, mike!

Please select the answer you believe to be correct
by pressing ↑/↓ arrows to move and ENTER to select

To start the game, press any key
(where's the 'any' key??)
```

App walkthrough - Logic of the code

Finer details

- Class
  - TriviaGame
- Functions
  - validate_name
  - difficulty
  - welcome_msg
  - play_game
  - player_score
  - corrections
  - what_next



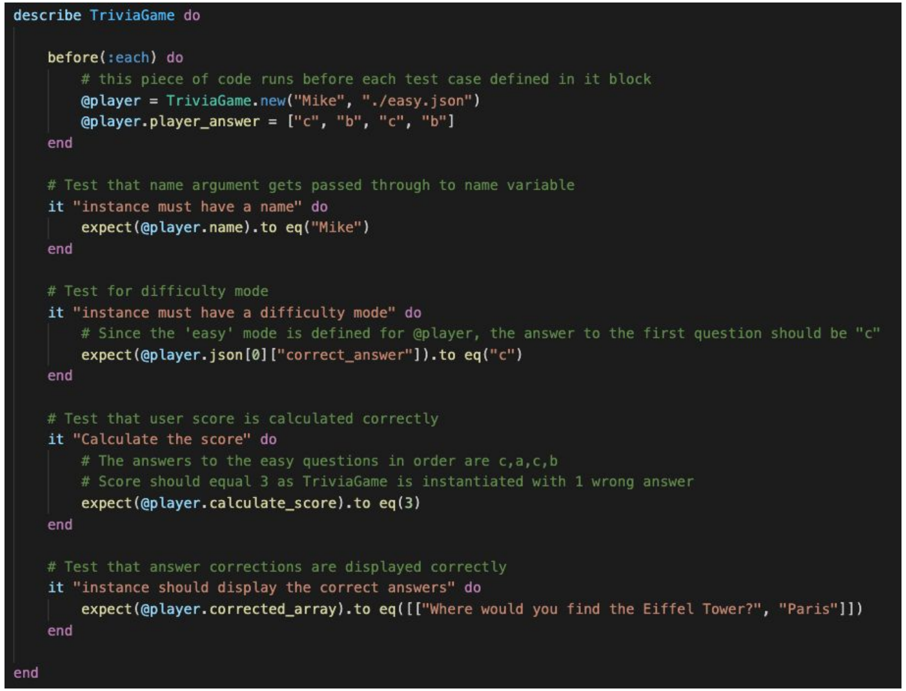
```ruby
class TriviaGame

    attr_reader :name, :score, :player_answer
    # initialise game with the player name, a score starting at zero, and an empty array for their
    answers
    def initialize(name, mode)
        @name = name
        @score = 0
        @player_answer = []
        # access and parse the JSON question file
        @@question_file = File.read(mode)
        @@json = JSON.parse(@@question_file)
        # @set the ascii font for headings
        @@ascii = Artii::Base.new :font => 'doom'
        @@games_played = 0
    end
```

Development & Build Process

Testing with RSpec

RSpec has been employed as the testing method of choice. Tests have been designed to cover four major features:

- The instance must have a name
- The instance must have a difficulty mode selected and load the corresponding JSON file
- Calculate the score based on number of correct answers
- Pull out questions the user answered wrong and provide the correct answer

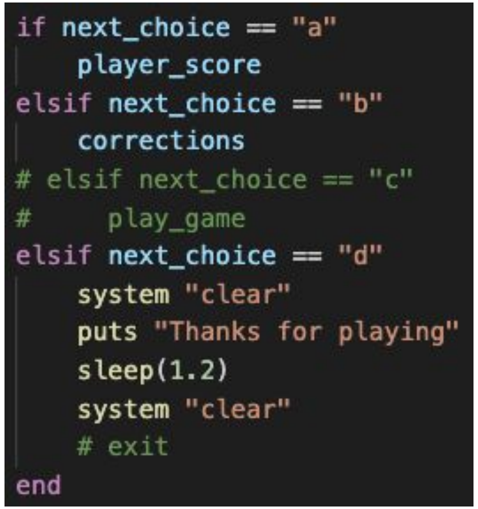


```ruby
describe TriviaGame do

    before(:each) do
        # this piece of code runs before each test case defined in it block
        @player = TriviaGame.new("Mike", "./easy.json")
        @player.player_answer = ["c", "b", "c", "b"]
    end

    # Test that name argument gets passed through to name variable
    it "instance must have a name" do
        expect(@player.name).to eq("Mike")
    end

    # Test for difficulty mode
    it "instance must have a difficulty mode" do
        # Since the 'easy' mode is defined for @player, the answer to the first question should be "c"
        expect(@player.json[0]["correct_answer"]).to eq("c")
    end

    # Test that user score is calculated correctly
    it "Calculate the score" do
        # The answers to the easy questions in order are c,a,c,b
        # Score should equal 3 as TriviaGame is instantiated with 1 wrong answer
        expect(@player.calculate_score).to eq(3)
    end

    # Test that answer corrections are displayed correctly
    it "instance should display the correct answers" do
        expect(@player.corrected_array).to eq([["Where would you find the Eiffel Tower?", "Paris"]])
    end

end
```

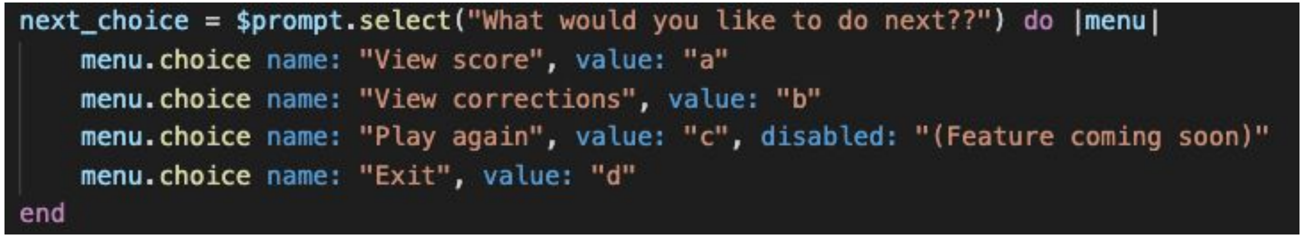
Development & Build Process

Challenges

- Constant improvements and their potential to break code
- Understanding Ruby docs
- Timeframe - importance of focused work and MVP
- Dealing with TTY-prompt return values - can't directly
- Dealing with variable scope



```ruby
if next_choice == "a"
    player_score
elsif next_choice == "b"
    corrections
# elsif next_choice == "c"
#     play_game
elsif next_choice == "d"
    system "clear"
    puts "Thanks for playing"
    sleep(1.2)
    system "clear"
    # exit
end
```



```ruby
next_choice = $prompt.select("What would you like to do next??") do |menu|
    menu.choice name: "View score", value: "a"
    menu.choice name: "View corrections", value: "b"
    menu.choice name: "Play again", value: "c", disabled: "(Feature coming soon)"
    menu.choice name: "Exit", value: "d"
end
```

Development & Build Process

Ethical issues

- Crediting Gem authors
- Took “inspiration” from CoderAcademy tutorial videos (validate name error handling)

Favourite parts

- The challenge to build and iterate so fast
- Moments when a concept finally clicks
- Finding interesting gems and figuring out novel ways to use them
- Ideating the ways I could further build upon it

THANK YOU