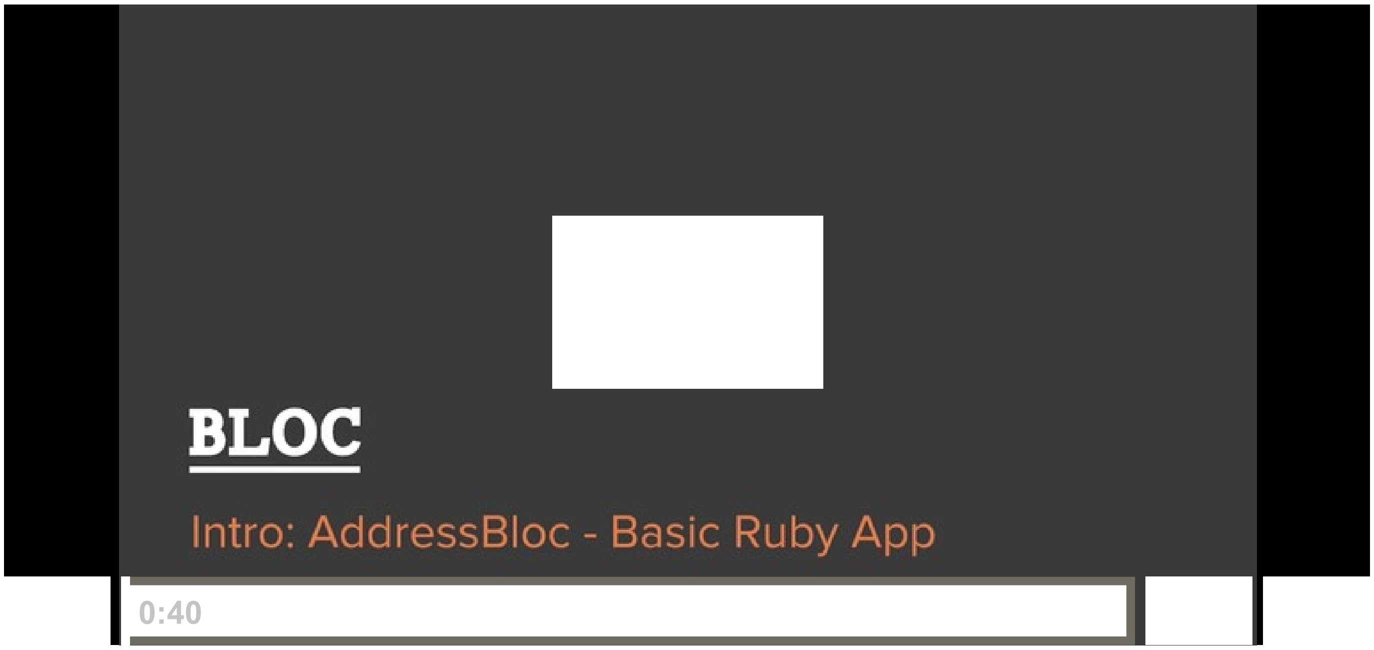
18 Address Bloc: Basic Ruby App

> “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.”
>
> — Yukihiro Matsumoto, the creator of Ruby

Introduction



We've spent the last several checkpoints learning Ruby in the browser, but now it's time to start using it locally. Over the next six checkpoints we'll build our first Ruby application, an address book application called Address Bloc. We'll learn how to run a Ruby program from the command line, create an interactive menu, represent the address book and entries with Ruby models, import entries from a file, search the address book, and testing.

IRB

The best tool to start working with Ruby locally is the Interactive Ruby Shell (IRB). IRB allows us to immediately execute Ruby commands on the fly.

> While working through the Ruby browser exercises you might have wondered how Ruby developers can memorize the massive collection of Ruby classes and methods. The secret is; they don't! If you ask a Ruby developer about an obscure class or method, you're likely to see them run IRB and start experimenting with it to remind themselves what it does.

Start IRB from the command line:

Terminal

```
$ irb
irb(main):001 >
```

> `irb(main):001 >` is the IRB prompt. `:001` is the current line number.

You can use IRB as a simple calculator:

Terminal

```
irb(main):001 > 3 + 3
 => 6
irb(main):002 >
```

`=> 6` is the value returned after we add `3` and `3` together.

Or to create and use variables:

Terminal

```
irb(main):002 > name = "John Jay"
 => "John Jay"
irb(main):003 > name.length
 => 8
irb(main):004 >
```

Or even to define and call methods. Let's create the **classic "Hello, World! program** in IRB:

Terminal

```
irb(main):004 > def hello
irb(main):005?>   puts "Hello, World!"
irb(main):006?>   end
 => :hello
 irb(main):007 > hello
 Hello, World!
  => nil
```

> `hello` doesn't return a value. Instead, it returns `nil`, as the `=> nil` line shows.

Let's watch a video that demonstrates how to use IRB from the command line:

Objects Solutions

Running a Ruby Program

IRB is a great tool for experimentation and learning, but when you exit IRB everything you wrote will be erased. To save code you'll write it in a Ruby file, which ends with an `.rb` extension. Exit IRB using the `exit` command and then create a new Ruby file:

Terminal

```
$ touch hello_world.rb
```

And then adding Ruby code inside of it:

hello_world.rb

```
+ def hello_world
+   puts "Hello, World!"
+ end
+
+ hello_world
```

That's it! We've created our first stand-alone Ruby program.

Ruby is an **interpreted language**, which means we can execute it directly, without needing to compile it first. Running it is simple as:

Terminal

```
$ ruby hello_world.rb
Hello, World!
```

Command-line Arguments

It's often useful to be able to pass arguments to our programs when we run them, via command-line arguments:

Terminal

```
$ ruby hello_world.rb arg1 arg2 arg3
```

Command-line arguments have many uses. They can be used to pass in the name of files to modify, provide options to change how the program is executed, or just pass in simple values to be used. Let's modify the hello world program so that can say hello to any number of people:

hello_world.rb

```
  def hello_world
  # #1
+   ARGV.each do |arg|
+     puts "Hello, #{arg}!"
+   end
-   puts "Hello, World!"
  end

 hello_world
```

#1, Command-line variables are stored in an array called `ARGV`. We can access and treat `ARGV` just like any other array.

Run the improved program and provide some people to say hello to via command-line arguments:

Terminal

```
$ ruby hello_world.rb Kermit Piggy Statler Waldorf Janice Fozzie Camilla
```

AddressBloc

Now that we've explored IRB and written our first Hello World program, it's time to start Address Bloc. Follow the **Git Checkpoint Workflow resource to create a repository** to create a repository on GitHub and clone it locally. Name it "address-bloc".

`cd` into `address-bloc` and create a new Git feature branch. See **Git Checkpoint Workflow: Before Each Checkpoint** for details.

Address Bloc will consist of multiple Ruby files, but for now we'll create the main Ruby program to run AddressBloc:

Terminal

```
$ touch address_bloc.rb
```

Open `address_bloc.rb` and add a message to welcome users to Address Bloc:

address_bloc.rb

```
+ puts "Welcome to AddressBloc!"
```

Run Address Bloc:

Terminal

```
$ ruby address_bloc.rb
Welcome to AddressBloc!
$
```

We've laid the foundation for our first Ruby program. In the next checkpoint, we'll add more functionally and create an interactive menu for Address Bloc users.

Git

Commit your checkpoint work in Git. See **Git Checkpoint Workflow: After Each Checkpoint** for details.

Recap

Concept	Description
IRB	IRB is a tool used to interactively execute Ruby expressions read from the standard input.
`ruby`	The `ruby` command invokes the Ruby interpreter to run Ruby programs from the command line.

Assignment | Discussion | Submission

Write a program named `greeting.rb` that takes multiple command-line arguments. The first argument should be the greeting to be used. The rest of the arguments should be the names of people to greet. The program should print out a greeting for each person. For example:

Terminal

```
$ ruby greeting.rb Hey Sterling Cheryl Lana
```

should output:

Terminal

```
Hey Sterling
Hey Cheryl
Hey Lana
```

and

Terminal

```
$ ruby greeting.rb Yo Cyril Archer Krieger
```

should output:

Terminal

```
Yo Cyril
Yo Archer
Yo Krieger
```

Create `greeting.rb` in your `code` directory. Once your assignment is complete, submit your code to your mentor via the Discussion tab.

Solution

Do not watch this video until after you've attempted to complete the assignment. If you struggle to complete the assignment, submit your best effort to your mentor *before watching a solution video*.

Address Bloc: Basic Ruby App Solution

assignment completed

?