ruby/debug

The best investment for your productivity

2022-09-09

shopify

Who Am I?

Who Am I?

- Stan Lo

Who Am I?

- Stan Lo
- Taiwanese 🇹🇼 Live in London 🇬🇧

shopify

Who Am I?

- Stan Lo
- Taiwanese 🇹🇼 Live in London 🇬🇧
- 8th RubyKaigi since 2015 🎉

shopify

Who Am I?

- Stan Lo
- Taiwanese 🇹🇼 Live in London 🇬🇧
- 8th RubyKaigi since 2015 🎉
- Work at Shopify's Ruby Developer Experience Team 🛠️

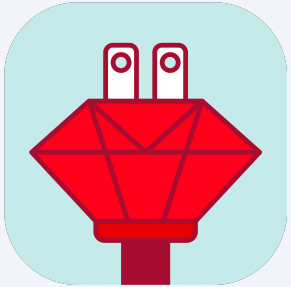
Who Am I?

- Stan Lo
- Taiwanese 🇹🇼 Live in London 🇬🇧
- 8th RubyKaigi since 2015 🎉
- Work at Shopify's Ruby Developer Experience Team 🛠️

shopify

Who Am I?

- Stan Lo
- Taiwanese 🇹🇼 Live in London 🇬🇧
- 8th RubyKaigi since 2015 🎉
- Work at Shopify's Ruby Developer Experience Team 🛠️



shopify

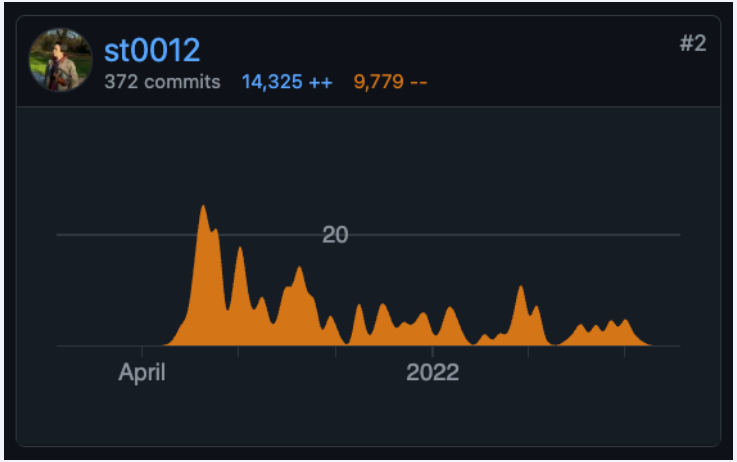
Who Am I?

- Stan Lo
- Taiwanese 🇹🇼 Live in London 🇬🇧
- 8th RubyKaigi since 2015 🎉
- Work at Shopify's Ruby Developer Experience Team 🛠️

shopify

Who Am I?

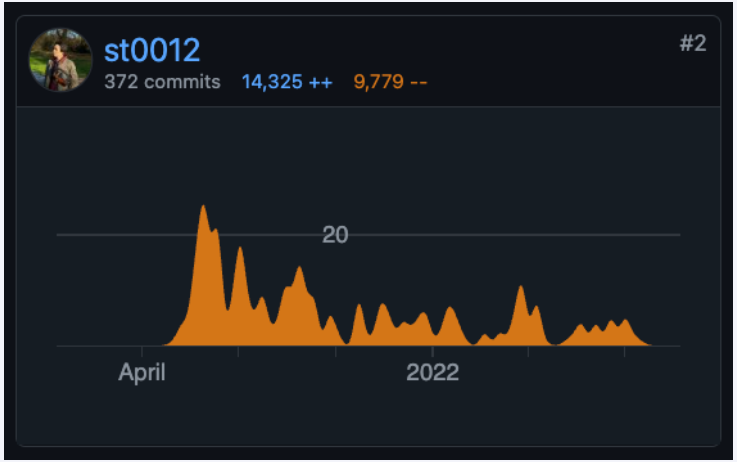
- Stan Lo
- Taiwanese 🇹🇼 Live in London 🇬🇧
- 8th RubyKaigi since 2015 🎉
- Work at Shopify's Ruby Developer Experience Team 🛠️
- Active contributor of ruby/debug ✨



shopify

Who Am I?

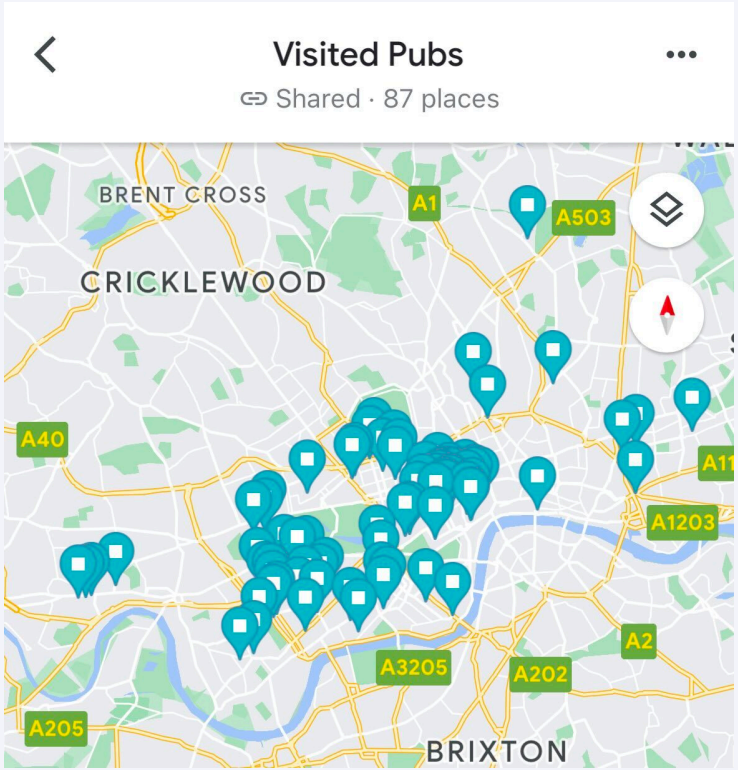
- Stan Lo
- Taiwanese 🇹🇼 Live in London 🇬🇧
- 8th RubyKaigi since 2015 🎉
- Work at Shopify's Ruby Developer Experience Team 🛠️
- Active contributor of ruby/debug ✨
- Maintainer of sentry-ruby 👷



shopify

Who Am I?

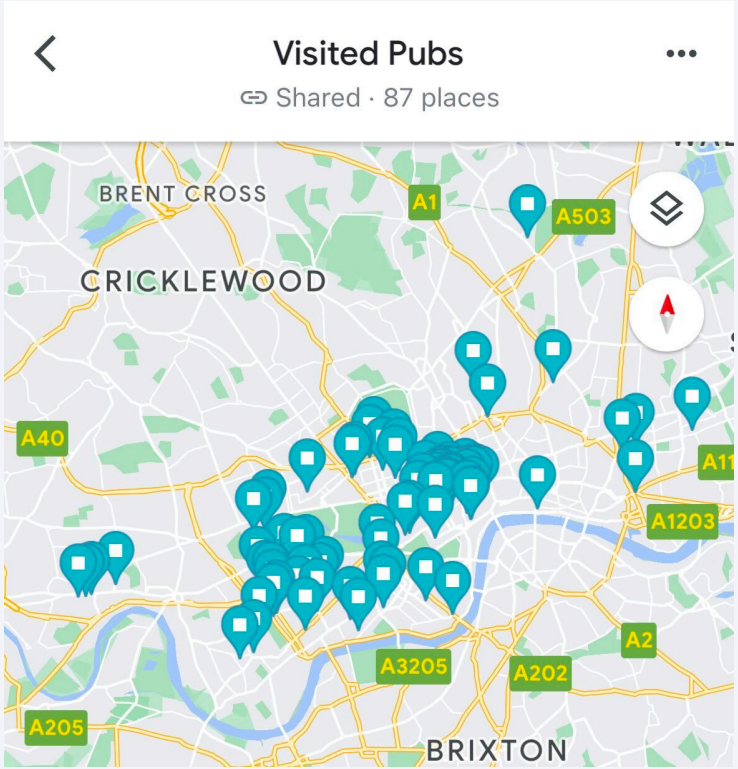
- Stan Lo
- Taiwanese 🇹🇼 Live in London 🇬🇧
- 8th RubyKaigi since 2015 🎉
- Work at Shopify's Ruby Developer Experience Team 🛠️
- Active contributor of ruby/debug ✨
- Maintainer of sentry-ruby 👷
- Explore London and UK's pubs. Current record: 87/100 🍻



shopify

Who Am I?

- Stan Lo
- Taiwanese 🇹🇼 Live in London 🇬🇧
- 8th RubyKaigi since 2015 🎉
- Work at Shopify's Ruby Developer Experience Team 🛠️
- Active contributor of ruby/debug ✨
- Maintainer of sentry-ruby 👷
- Explore London and UK's pubs. Current record: 87/100 🍻
- Github: @st0012 Twitter: @_st0012



shopify

Why is debugging important?

~~Why is debugging important?~~

shopify

Do we keep our debugging skills sharp?

~~Why is debugging important?~~

Do we keep our debugging skills sharp?

shopify

I didn’t

shopify

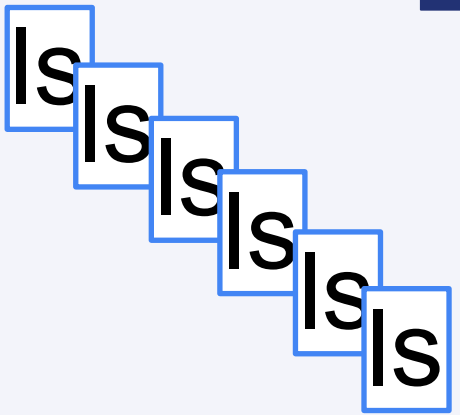
I didn’t

binding.pry

binding.pry

binding.pry

binding.pry



I didn’t

binding.pry

ls

ls

ls

ls

ls

ls

binding.pry

I didn't

binding.pry

binding.pry

show-source

show-source

show-source

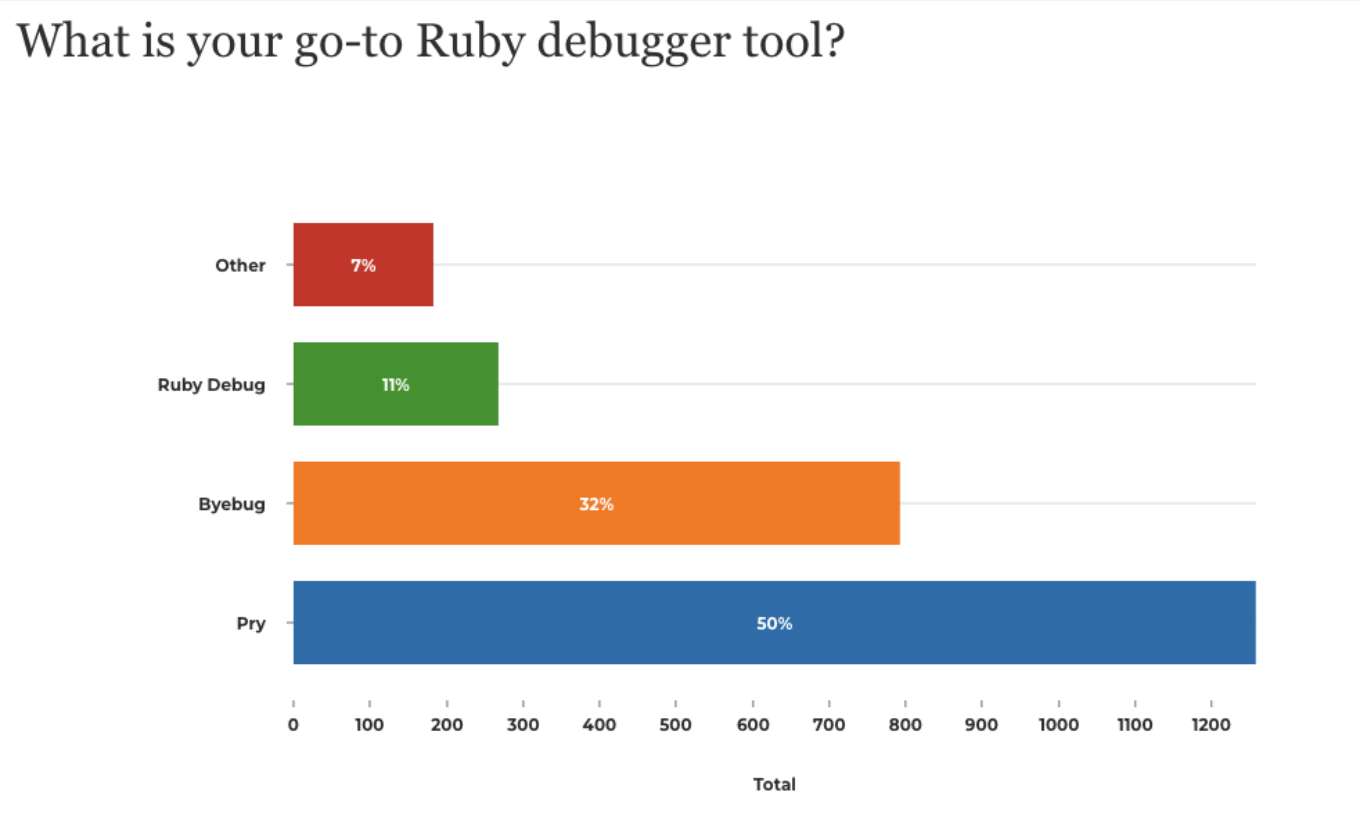
show-source

show-source

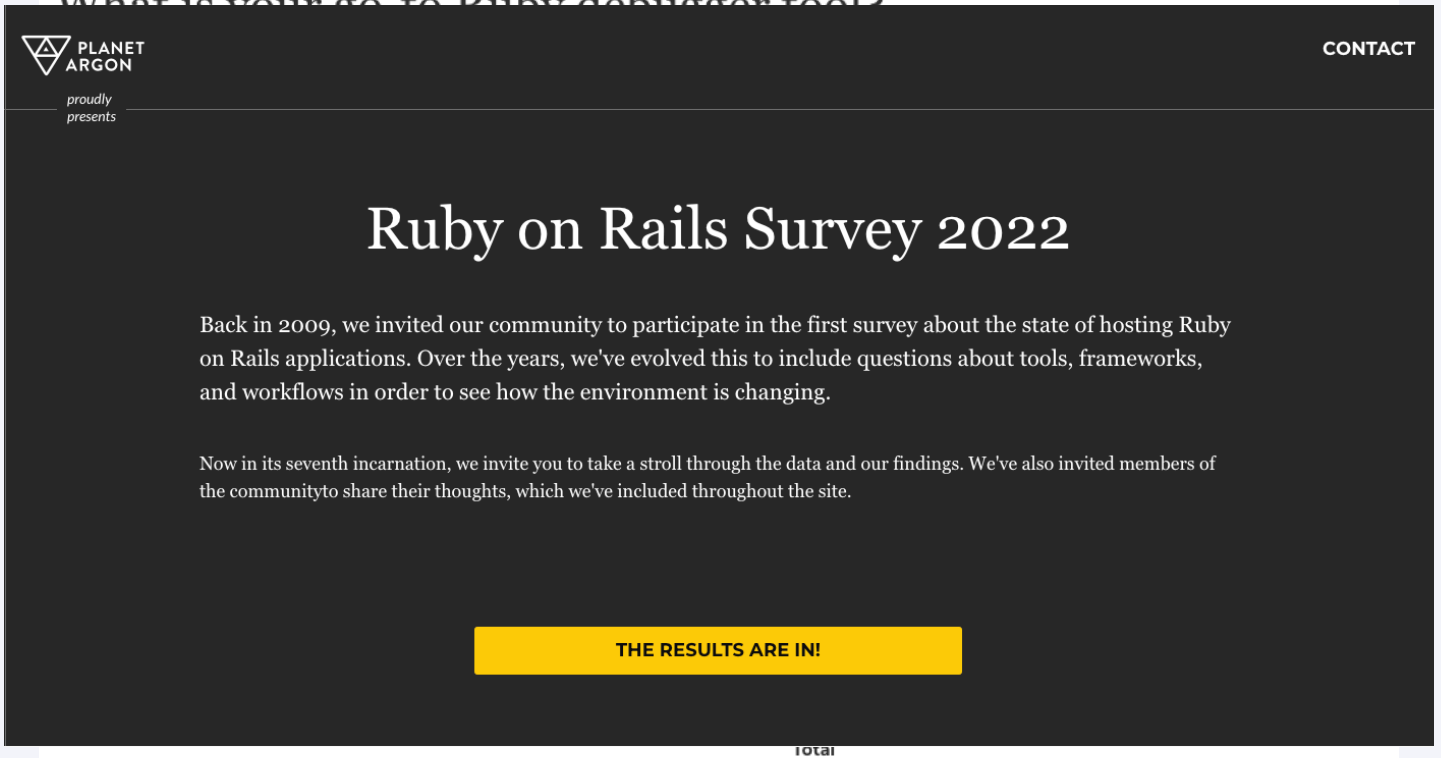
show-source

shopify

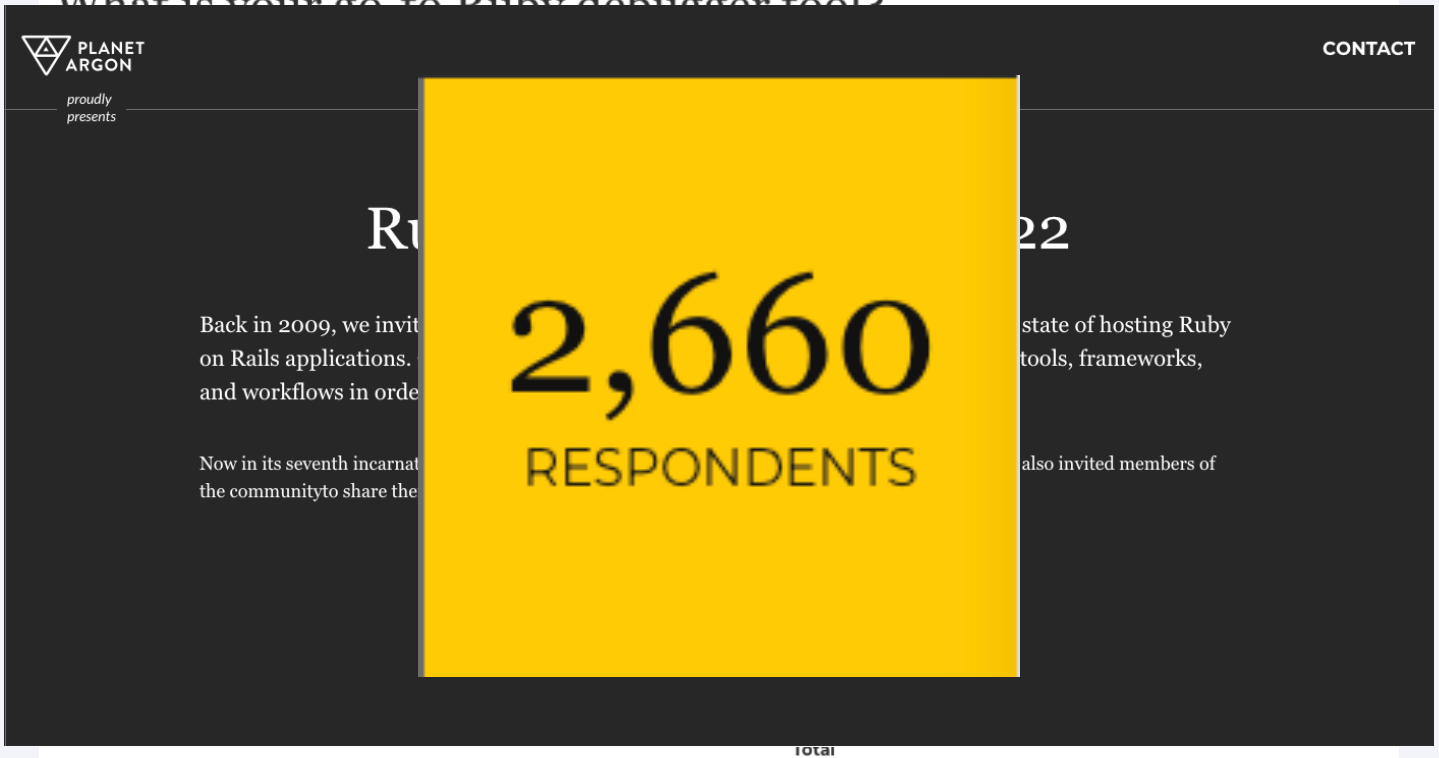
What do Ruby developers use for debugging?



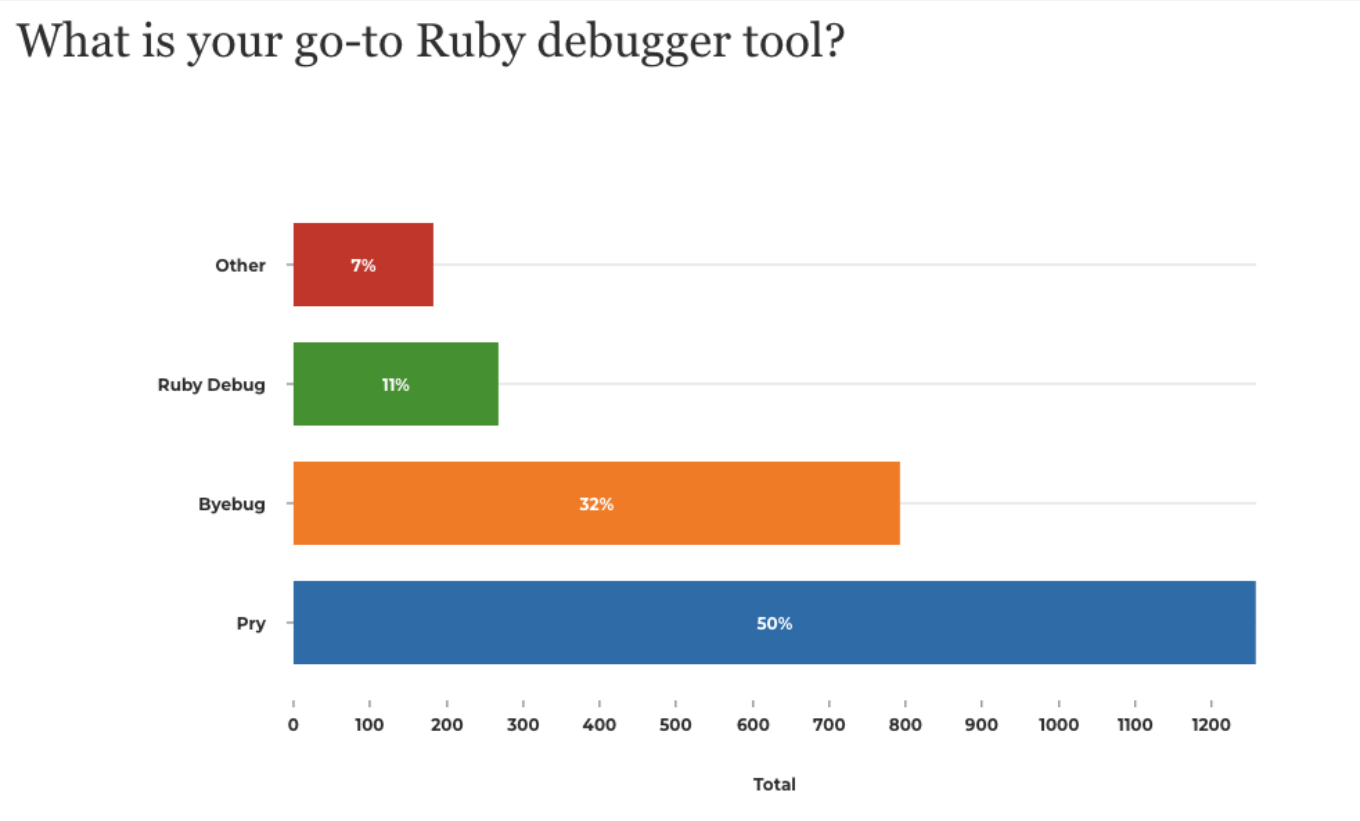
https://rails-hosting.com/2022



https://rails-hosting.com/2022

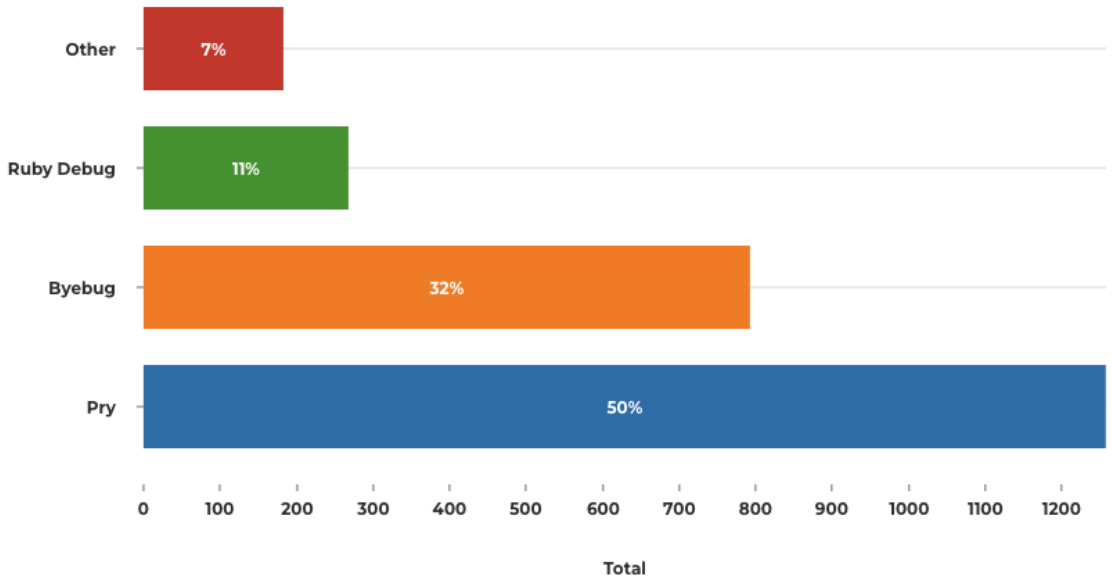


https://rails-hosting.com/2022



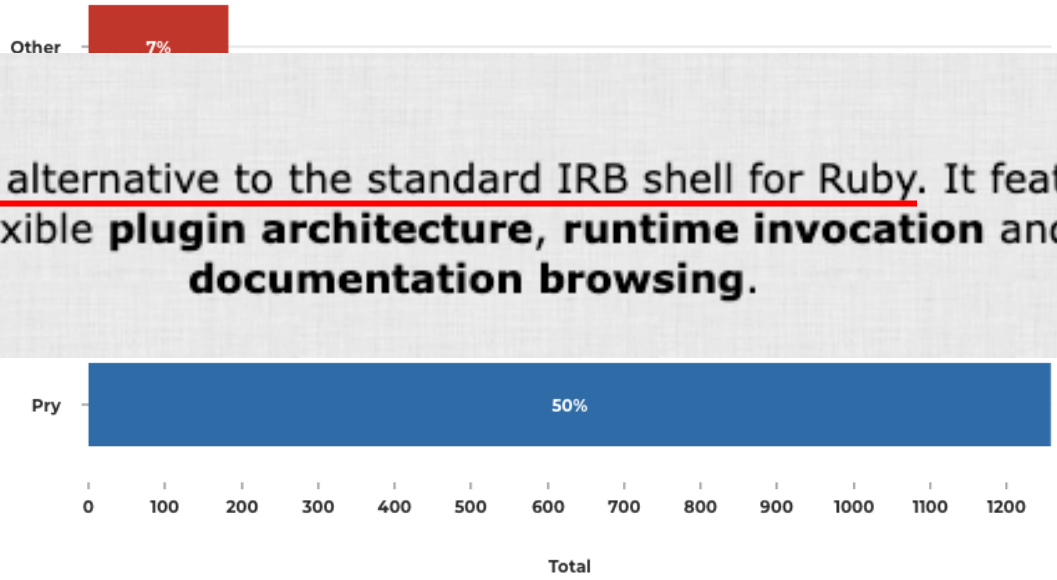
https://rails-hosting.com/2022

What is your go-to Ruby debugger tool?



https://rails-hosting.com/2022

What is your go-to Ruby debugger tool?



Pry is a **powerful** alternative to the standard IRB shell for Ruby. It features **syntax highlighting**, a flexible **plugin architecture**, **runtime invocation** and **source and documentation browsing**.

https://rails-hosting.com/2022

Common Debugger Features

- Step-debugging
- Frame Navigation
- Breakpoint Commands

Common Debugger Features

- Step-debugging
- Frame Navigation
- Breakpoint Commands
- Scriptable Breakpoint (ruby/debug only)

Example

```ruby
require "debug"
load "lib.rb"

binding.b
result = foo(100)

if result == 202
  puts("foo works as expected")
else
  puts("foo returned incorrect value: #{result}")
end
```

main.rb

Example

```ruby
require "debug"
load "lib.rb"

binding.b
result = foo(100)

if result == 202
  puts("foo works as expected")
else
  puts("foo returned incorrect value: #{result}")
end
```

main.rb

Example

```ruby
require "debug"
load "lib.rb"

binding.b
result = foo(100)

if result == 202
  puts("foo works as expected")
else
  puts("foo returned incorrect value: #{result}")
end
```

main.rb

Example

```ruby
require "debug"
load "lib.rb"

binding.b
result = foo(100)

if result == 202
  puts("foo works as expected")
else
  puts("foo returned incorrect value: #{result}")
end
```

main.rb

Example

```ruby
require "debug"
load "lib.rb"

binding.b
result = foo(100)

if result == 202
  puts("foo works as expected")
else
  puts("foo returned incorrect value: #{result}")
end
```

main.rb

Example

```ruby
require "debug" 👈 require the debugger
load "lib.rb"

binding.b
result = foo(100)

if result == 202
  puts("foo works as expected")
else
  puts("foo returned incorrect value: #{result}")
end
```

main.rb

Example

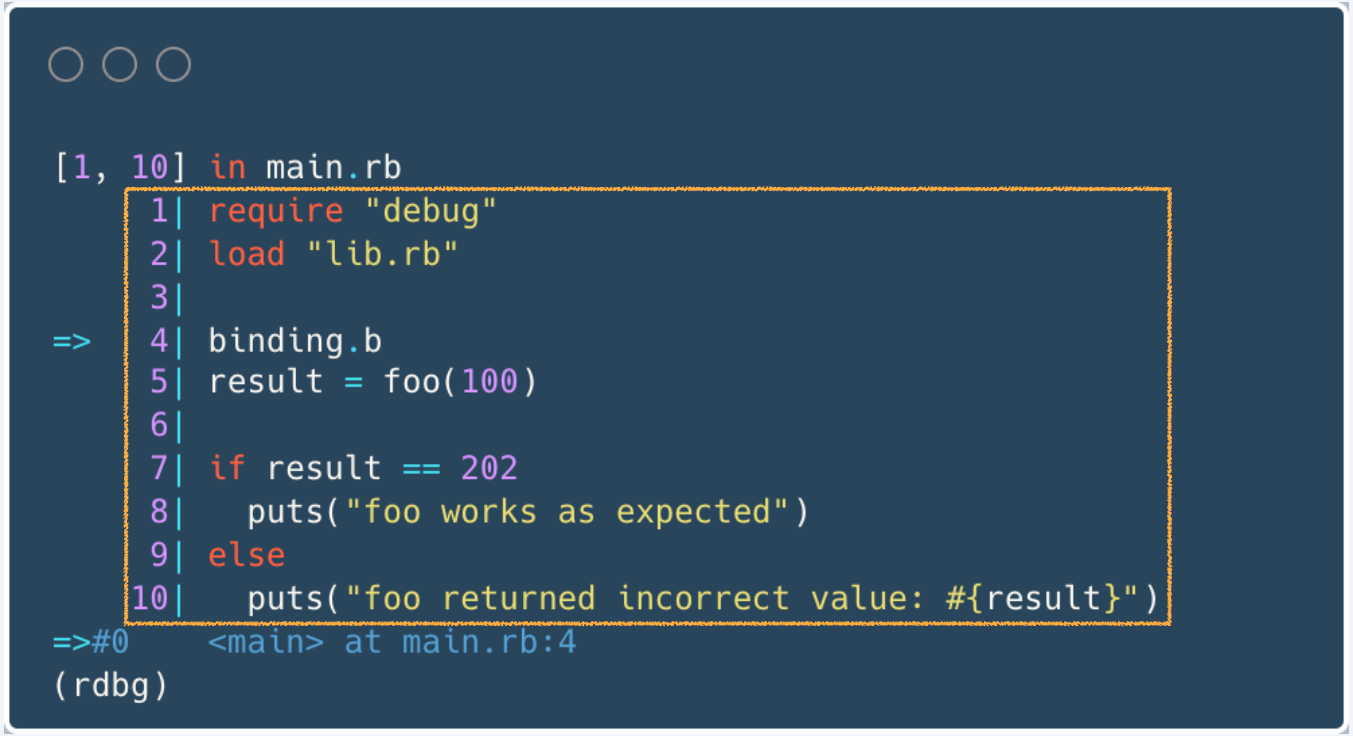
```ruby
require "debug" 👈 require the debugger
load "lib.rb"

binding.b 👈 set a breakpoint
result = foo(100)

if result == 202
  puts("foo works as expected")
else
  puts("foo returned incorrect value: #{result}")
end
```

main.rb

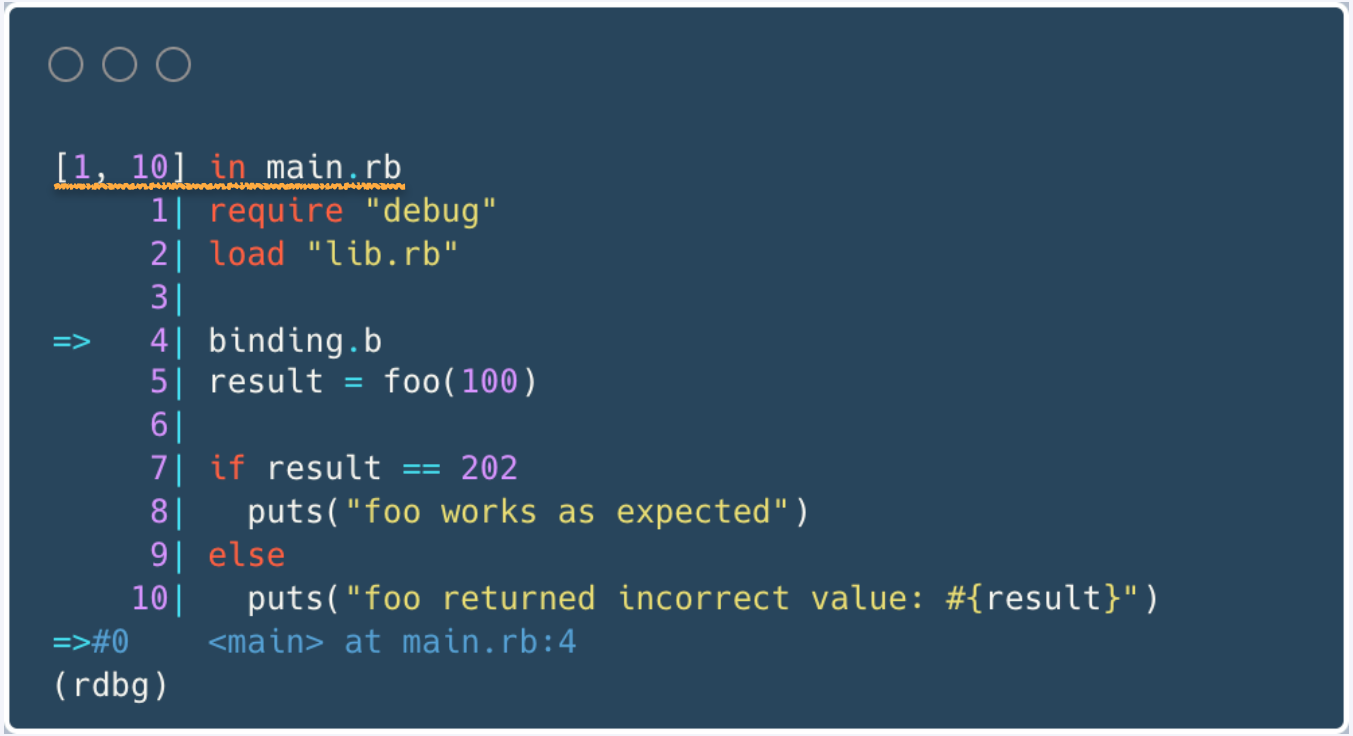
Entering a debugging session

```
[1, 10] in main.rb
     1| require "debug"
     2| load "lib.rb"
     3|
=>   4| binding.b
     5| result = foo(100)
     6|
     7| if result == 202
     8|   puts("foo works as expected")
     9| else
    10|   puts("foo returned incorrect value: #{result}")
=>#0    <main> at main.rb:4
(rdbg)
```

Entering a debugging session



Entering a debugging session



Entering a debugging session

```
[1, 10] in main.rb
     1| require "debug"
     2| load "lib.rb"
     3|
=>   4| binding.b
     5| result = foo(100)
     6|
     7| if result == 202
     8|   puts("foo works as expected")
     9| else
    10|   puts("foo returned incorrect value: #{result}")
=>#0    <main> at main.rb:4
(rdbg)
```

Entering a debugging session

```
[1, 10] in main.rb
     1| require "debug"
     2| load "lib.rb"
     3|
=>   4| binding.b
     5| result = foo(100)
     6|
     7| if result == 202
     8|   puts("foo works as expected")
     9| else
    10|   puts("foo returned incorrect value: #{result}")
=>#0    <main> at main.rb:4
(rdbg)
```

Entering a debugging session

```
[1, 10] in main.rb
     1| require "debug"
     2| load "lib.rb"
     3|
=>   4| binding.b
     5| result = foo(100)
     6|
     7| if result == 202
     8|   puts("foo works as expected")
     9| else
    10|   puts("foo returned incorrect value: #{result}")
=>#0    <main> at main.rb:4
(rdbg)
```

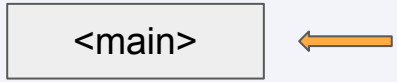
Step Debugging

Micro-managing our program

shopify

Step Debugging

```
$ ruby main.rb
[1, 10] in main.rb
     1| require "debug"
     2| load "lib.rb"
     3|
=>   4| binding.b
     5| result = foo(100)
     6|
     7| if result == 202
     8|   puts("foo works as expected")
     9| else
    10|   puts("foo returned incorrect value: #{result}")
=>#0    <main> at main.rb:4
(rdbg)
```



Step Debugging

```
$ ruby main.rb
[1, 10] in main.rb
     1| require "debug"
     2| load "lib.rb"
     3|
=>   4| binding.b
     5| result = foo(100)
     6|
     7| if result == 202
     8|   puts("foo works as expected")
     9| else
    10|   puts("foo returned incorrect value: #{result}")
=>#0    <main> at main.rb:4
(rdbg)
```

step

<main>

Step Debugging

```
$ ruby main.rb
[1, 10] in main.rb
     1| require "debug"
     2| load "lib.rb"
     3|
=>   4| binding.b
     5| result = foo(100)
     6|
     7| if result == 202
     8|   puts("foo works as expected")
     9| else
    10|   puts("foo returned incorrect value: #{result}")
=>#0    <main> at main.rb:4
(rdbg)
```

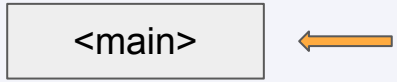
step

next

<main>

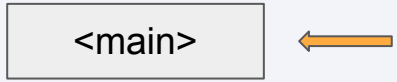
Step Debugging

```
$ ruby main.rb
[1, 10] in main.rb
     1| require "debug"
     2| load "lib.rb"
     3|
=>   4| binding.b
     5| result = foo(100)
     6|
     7| if result == 202
     8|   puts("foo works as expected")
     9| else
    10|   puts("foo returned incorrect value: #{result}")
=>#0    <main> at main.rb:4
(rdbg)
```



Step Debugging

```
(rdbg) s    # step command
[1, 10] in main.rb
     1| require "debug"
     2| load "lib.rb"
     3|
     4| binding.b
=>   5| result = foo(100)
     6|
     7| if result == 202
     8|   puts("foo works as expected")
     9| else
    10|   puts("foo returned incorrect value: #{result}")
=>#0    <main> at main.rb:5
(rdbg)
```



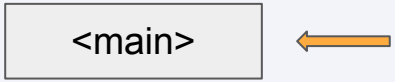
Step Debugging

```
(rdbg) s    # step command
[1, 10] in main.rb
     1| require "debug"
     2| load "lib.rb"
     3|
     4| binding.b
=>   5| result = foo(100)
     6|
     7| if result == 202
     8|   puts("foo works as expected")
     9| else
    10|   puts("foo returned incorrect value: #{result}")
=>#0    <main> at main.rb:5
(rdbg)
```

<main>

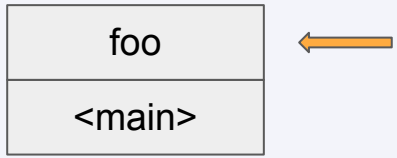
Step Debugging

```
(rdbg) s    # step command
[1, 10] in main.rb
     1| require "debug"
     2| load "lib.rb"
     3|
     4| binding.b
=>   5| result = foo(100)
     6|
     7| if result == 202
     8|   puts("foo works as expected")
     9| else
    10|   puts("foo returned incorrect value: #{result}")
=>#0    <main> at main.rb:5
(rdbg)
```



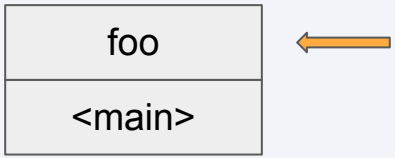
Step Debugging

```
(rdbg) s    # step command
[1, 10] in lib.rb
     1| def foo(n)
=>   2|   bar(n)
     3| end
     4|
     5| def bar(n)
     6|   baz(n)
     7| end
     8|
     9| def baz(n)
    10|   num = plus_1(n)
=>#0    Object#foo(n=100) at lib.rb:2
  #1    <main> at main.rb:5
(rdbg)
```



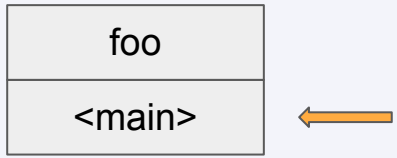
Step Debugging

```
(rdbg) s    # step command
[1, 10] in lib.rb
     1| def foo(n)
=>   2|   bar(n)
     3| end
     4|
     5| def bar(n)
     6|   baz(n)
     7| end
     8|
     9| def baz(n)
    10|   num = plus_1(n)
=>#0    Object#foo(n=100) at lib.rb:2
  #1    <main> at main.rb:5
(rdbg)
```



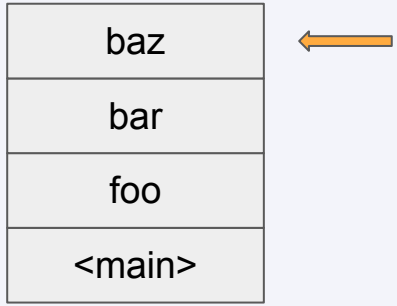
Step Debugging

```
(rdbg) s    # step command
[1, 10] in lib.rb
     1| def foo(n)
=>   2|   bar(n)
     3| end
     4|
     5| def bar(n)
     6|   baz(n)
     7| end
     8|
     9| def baz(n)
    10|   num = plus_1(n)
=>#0    Object#foo(n=100) at lib.rb:2
  #1    <main> at main.rb:5
(rdbg)
```



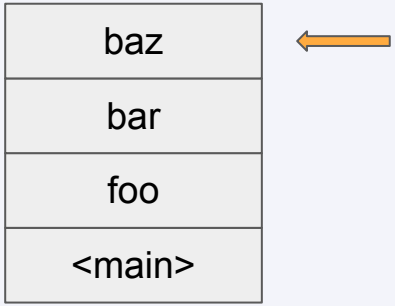
Step Debugging

```
(rdbg) s 2    # step command
[5, 14] in lib.rb
     5| def bar(n)
     6|   baz(n)
     7| end
     8|
     9| def baz(n)
=>  10|   num = plus_1(n)
    11|   double(num)
    12| end
    13|
    14| def plus_1(n)
=>#0    Object#baz(n=100) at lib.rb:10
  #1    Object#bar(n=100) at lib.rb:6
  # and 2 frames (use `bt' command for all frames)
(rdbg)
```



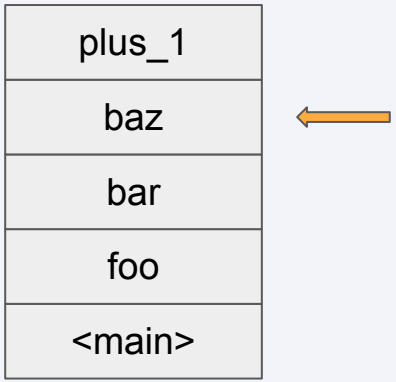
Step Debugging

```
(rdbg) s 2    # step command
[5, 14] in lib.rb
     5| def bar(n)
     6|   baz(n)
     7| end
     8|
     9| def baz(n)
=>  10|   num = plus_1(n)
    11|   double(num)
    12| end
    13|
    14| def plus_1(n)
=>#0    Object#baz(n=100) at lib.rb:10
  #1    Object#bar(n=100) at lib.rb:6
  # and 2 frames (use `bt' command for all frames)
(rdbg)
```



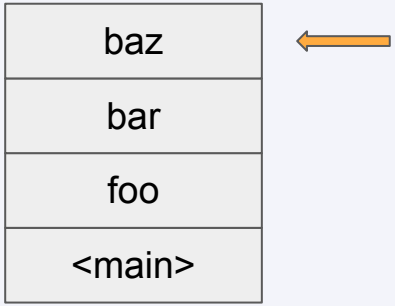
Step Debugging

```
(rdbg) s 2    # step command
[5, 14] in lib.rb
     5| def bar(n)
     6|   baz(n)
     7| end
     8|
     9| def baz(n)
=>  10|   num = plus_1(n)
    11|   double(num)
    12| end
    13|
    14| def plus_1(n)
=>#0    Object#baz(n=100) at lib.rb:10
  #1    Object#bar(n=100) at lib.rb:6
  # and 2 frames (use `bt' command for all frames)
(rdbg)
```



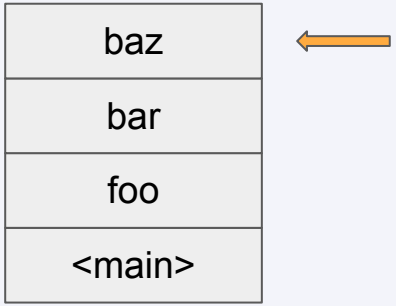
Step Debugging

```
(rdbg) s 2    # step command
[5, 14] in lib.rb
     5| def bar(n)
     6|   baz(n)
     7| end
     8|
     9| def baz(n)
=>  10|   num = plus_1(n)
    11|   double(num)
    12| end
    13|
    14| def plus_1(n)
=>#0    Object#baz(n=100) at lib.rb:10
  #1    Object#bar(n=100) at lib.rb:6
  # and 2 frames (use `bt' command for all frames)
(rdbg)
```



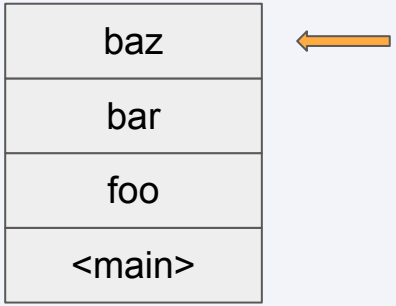
Step Debugging

```
(rdbg) n    # next command
[6, 15] in lib.rb
     6|   baz(n)
     7| end
     8|
     9| def baz(n)
    10|   num = plus_1(n)
=>  11|   double(num)
    12| end
    13|
    14| def plus_1(n)
    15|   n - 1
=>#0    Object#baz(n=100) at lib.rb:11
  #1    Object#bar(n=100) at lib.rb:6
  # and 2 frames (use `bt' command for all frames)
(rdbg)
```



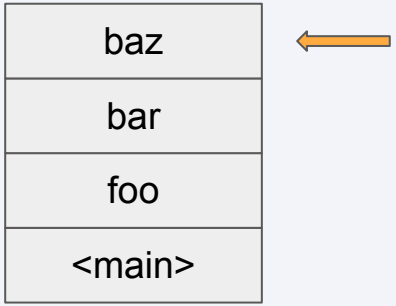
Step Debugging

```
     9| def baz(n)
    10|   num = plus_1(n)
=>  11|   double(num)
    12| end
    13|
    14| def plus_1(n)
    15|   n - 1
=>#0    Object#baz(n=100) at lib.rb:11
  #1    Object#bar(n=100) at lib.rb:6
  # and 2 frames (use `bt' command for all frames)
(rdbg) i    # info command
%self = main
n = 100
num = 99
(rdbg)
```



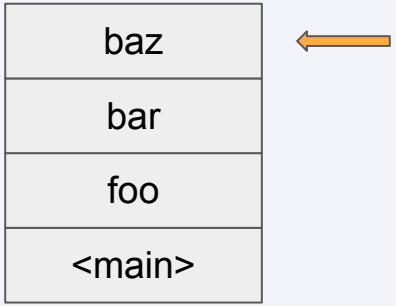
Step Debugging

```
     9| def baz(n)
    10|   num = plus_1(n)
=>  11|   double(num)
    12| end
    13|
    14| def plus_1(n)
    15|   n - 1
=>#0    Object#baz(n=100) at lib.rb:11
  #1    Object#bar(n=100) at lib.rb:6
  # and 2 frames (use `bt' command for all frames)
(rdbg) i    # info command
%self = main
n = 100
num = 99
(rdbg)
```



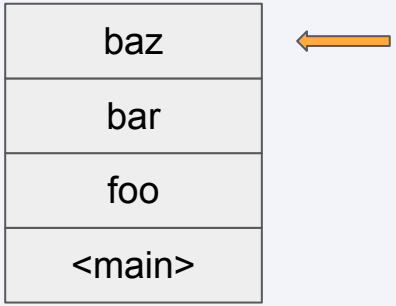
Step Debugging

```
     9| def baz(n)
    10|   num = plus_1(n)
=>  11|   double(num)
    12| end
    13|
    14| def plus_1(n)
    15|   n - 1
=>#0    Object#baz(n=100) at lib.rb:11
  #1    Object#bar(n=100) at lib.rb:6
  # and 2 frames (use `bt' command for all frames)
(rdbg) i    # info command
%self = main
n = 100
num = 99
(rdbg)
```



Step Debugging

```
     9| def baz(n)
    10|   num = plus_1(n)
=>  11|   double(num)
    12| end
    13|
    14| def plus_1(n)
    15|   n - 1
=>#0    Object#baz(n=100) at lib.rb:11
  #1    Object#bar(n=100) at lib.rb:6
  # and 2 frames (use `bt' command for all frames)
(rdbg) i    # info command
%self = main
n = 100
num = 99
(rdbg)
```



Step Debugging

Step Debugging

```
(rdbg) num
99
(ruby) methods.count
66
(rdbg) ls    # outline command
Object.methods: inspect  to_s
locals: n  num
(rdbg)
```

Step Debugging

```
(rdbg) num
99
(ruby) methods.count
66
(rdbg) ls    # outline command
Object.methods: inspect  to_s
locals: n  num
(rdbg)
```

Step Debugging

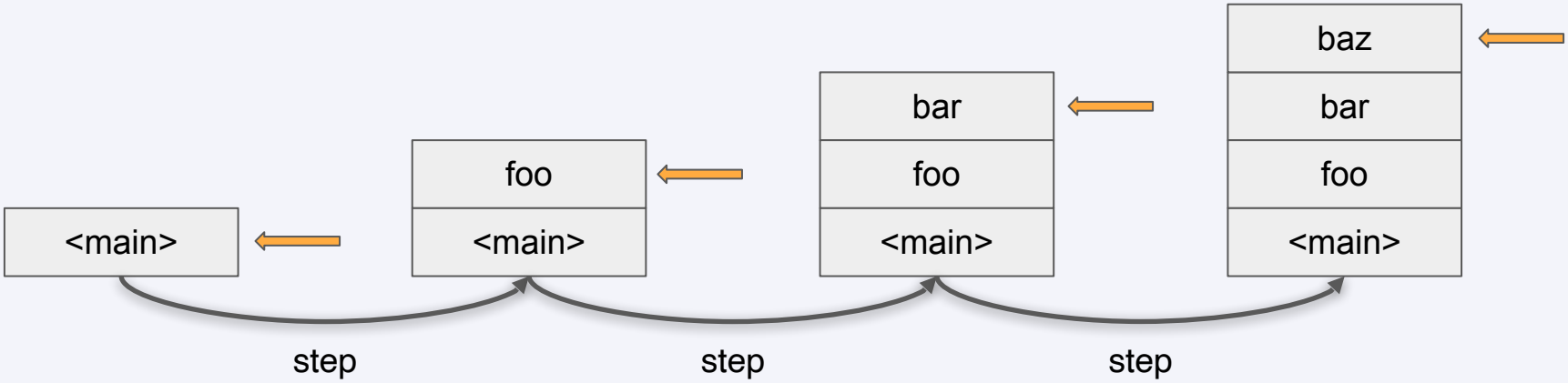
```
(rdbg) num
99
(ruby) methods.count
66
(rdbg) ls    # outline command
Object.methods: inspect  to_s
locals: n  num
(rdbg)
```

Step Debugging

```
(rdbg) num
99
(ruby) methods.count
66
(rdbg) ls    # outline command
Object.methods: inspect  to_s
locals: n  num
(rdbg)
```

Step Debugging

Step Debugging



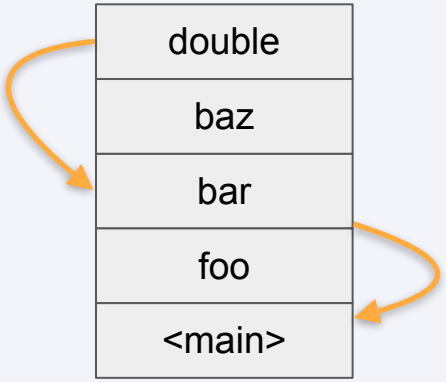
Frame Navigation

The best friend of step-debugging

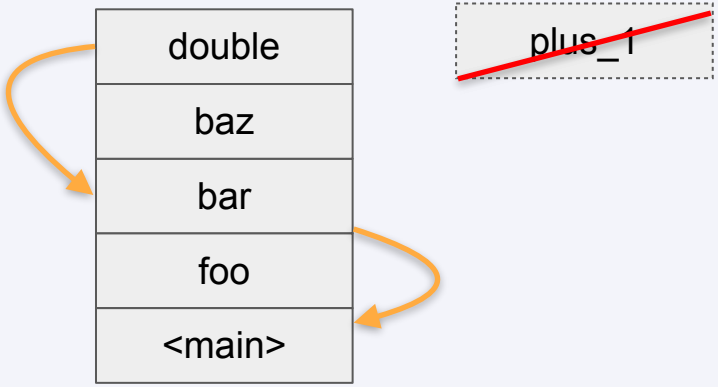
shopify

Frame Navigation

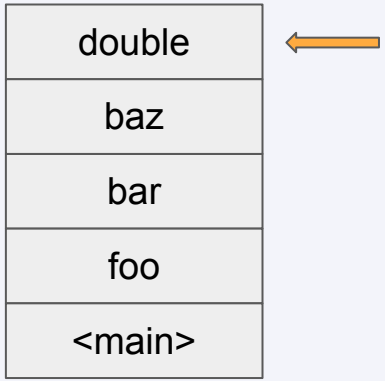
Frame Navigation



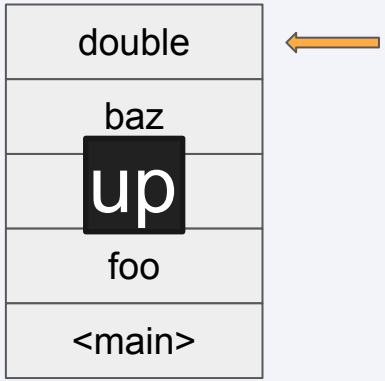
Frame Navigation



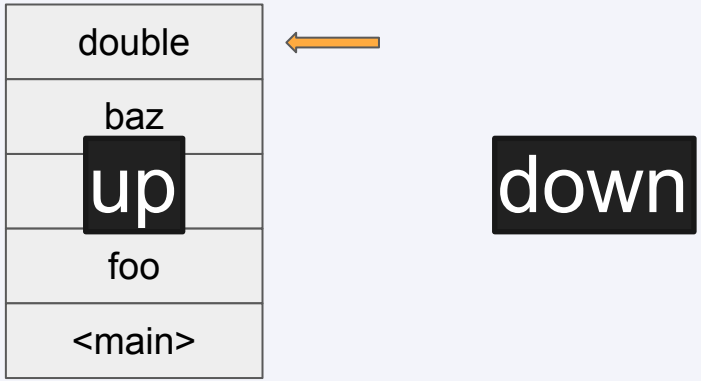
Frame Navigation



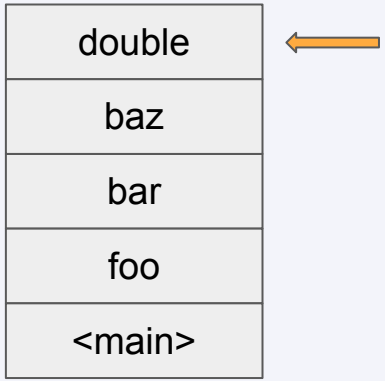
Frame Navigation



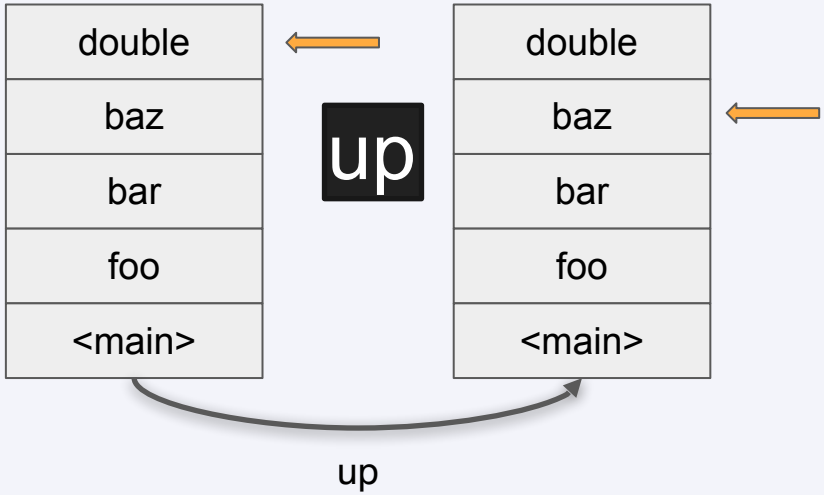
Frame Navigation



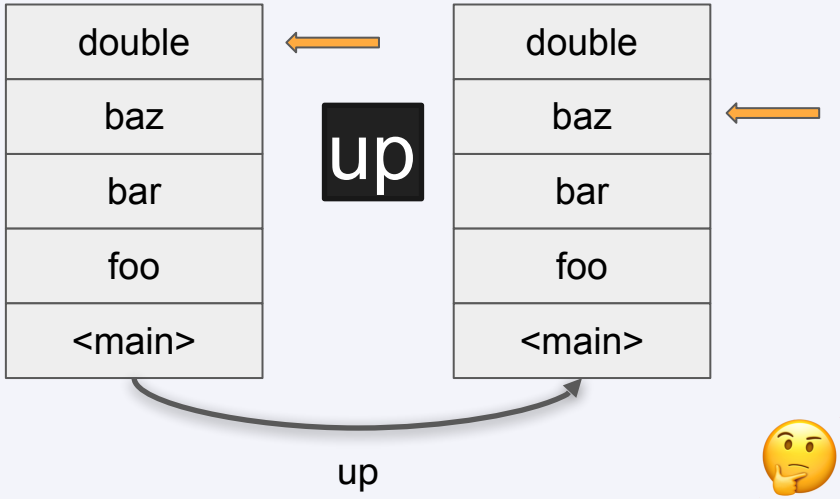
Frame Navigation



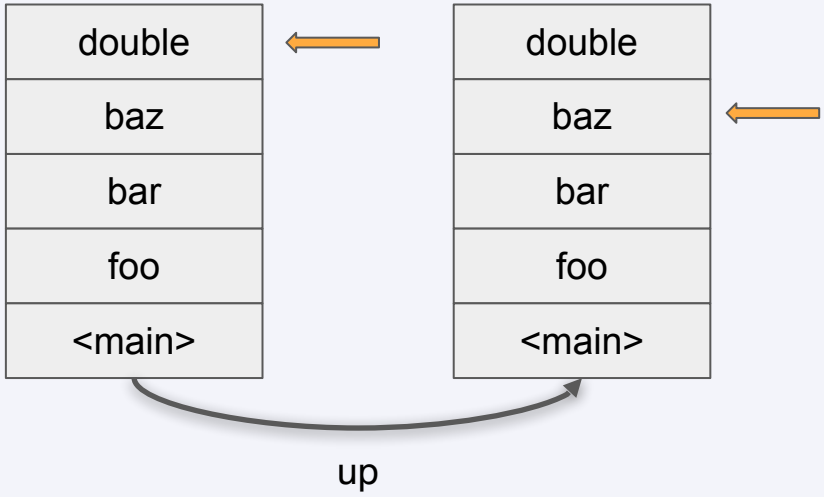
Frame Navigation



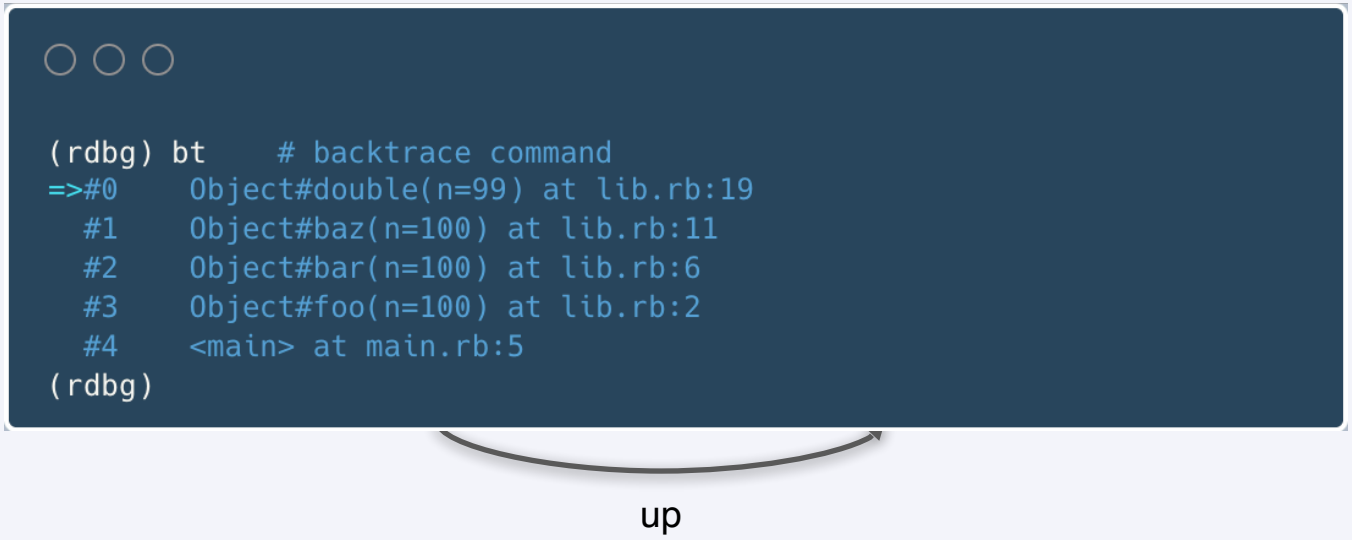
Frame Navigation



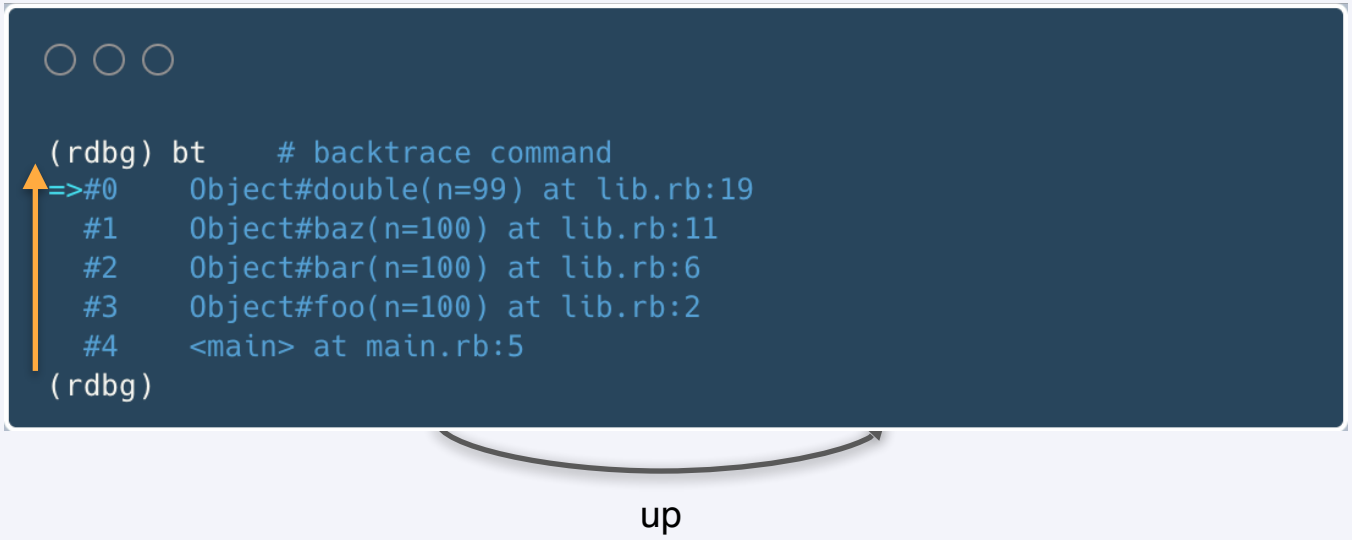
Frame Navigation



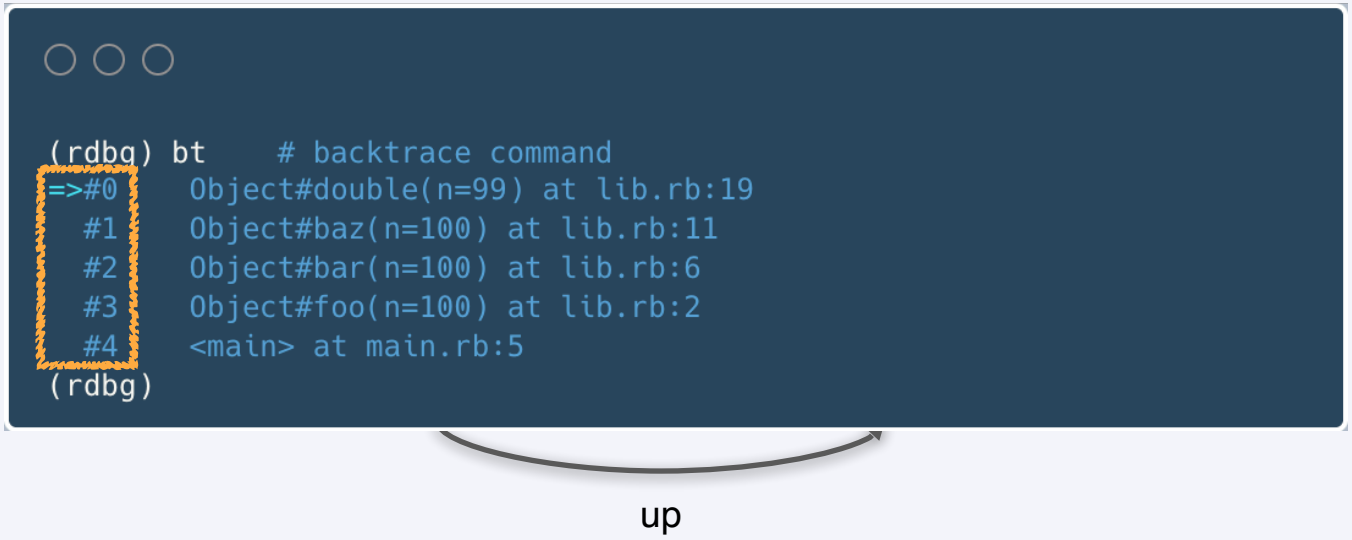
Frame Navigation



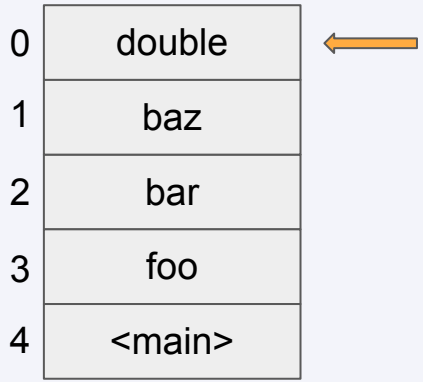
Frame Navigation



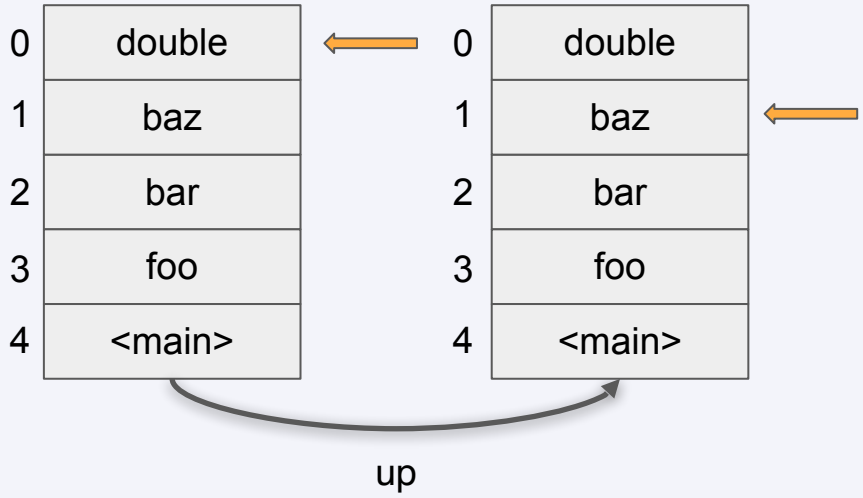
Frame Navigation



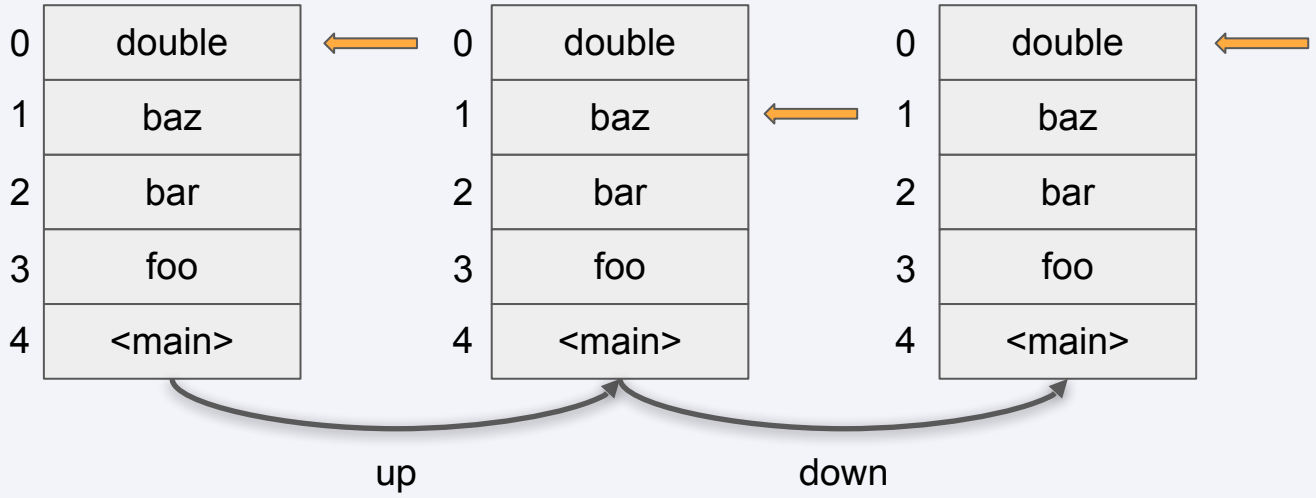
Frame Navigation



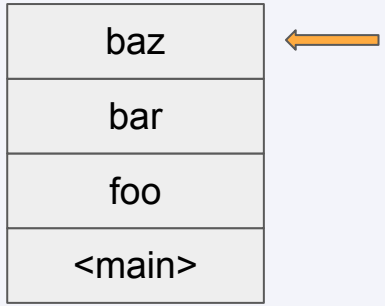
Frame Navigation



Frame Navigation

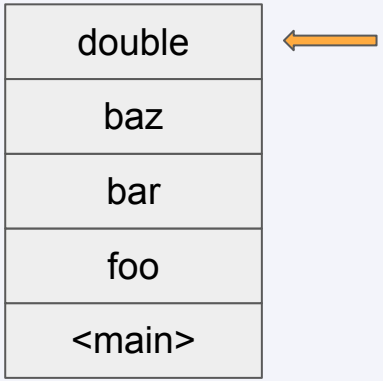


Frame Navigation



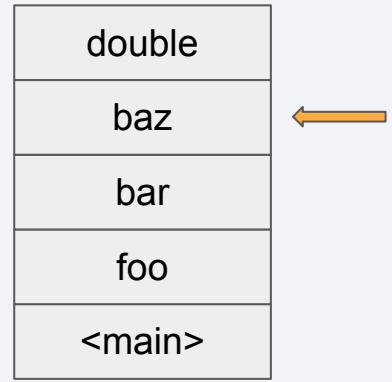
Frame Navigation

```
(rdbg) s    # step command
[14, 20] in lib.rb
    14| def plus_1(n)
    15|   n - 1
    16| end
    17|
    18| def double(n)
=>  19|   n * 2
    20| end
=>#0    Object#double(n=99) at lib.rb:19
  #1    Object#baz(n=100) at lib.rb:11
  # and 3 frames (use `bt' command for all frames)
(rdbg)
```



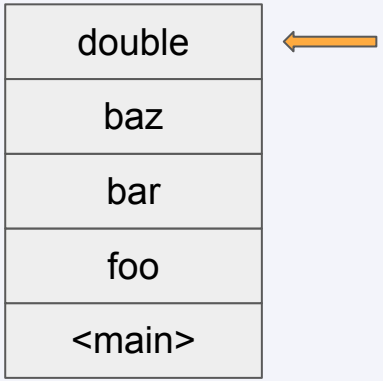
Frame Navigation

```
(rdbg) up    # command
=>  11|   double(num)
=>#1    Object#baz(n=100) at lib.rb:11
(rdbg) list    # command
     6|   baz(n)
     7| end
     8|
     9| def baz(n)
    10|   num = plus_1(n)
=>  11|   double(num)
    12| end
    13|
    14| def plus_1(n)
    15|   n - 1
(rdbg)
```



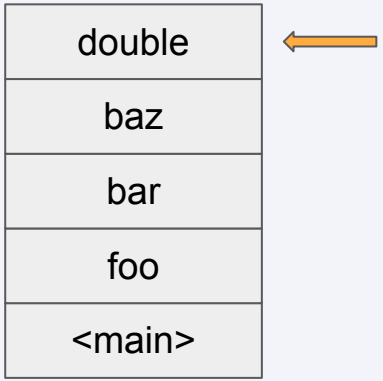
Frame Navigation

```
(rdbg) down    # command
=>  19|   n * 2
=>#0    Object#double(n=99) at lib.rb:19
(rdbg) list    # command
    14| def plus_1(n)
    15|   n - 1
    16| end
    17|
    18| def double(n)
=>  19|   n * 2
    20| end
(rdbg)
```



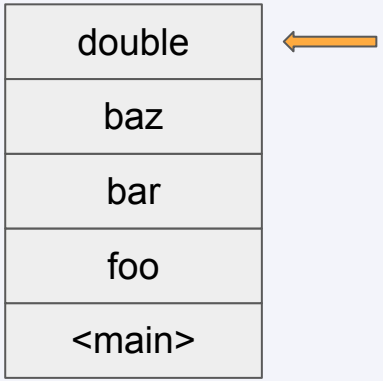
Frame Navigation

```
(rdbg) bt    # backtrace command
=>#0    Object#double(n=99) at lib.rb:19
  #1    Object#baz(n=100) at lib.rb:11
  #2    Object#bar(n=100) at lib.rb:6
  #3    Object#foo(n=100) at lib.rb:2
  #4    <main> at main.rb:5
(rdbg)
```



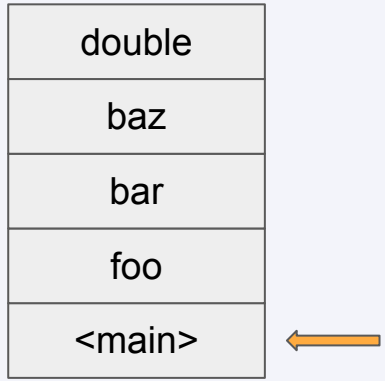
Frame Navigation

```
(rdbg) bt    # backtrace command
=>#0    Object#double(n=99) at lib.rb:19
  #1    Object#baz(n=100) at lib.rb:11
  #2    Object#bar(n=100) at lib.rb:6
  #3    Object#foo(n=100) at lib.rb:2
  #4    <main> at main.rb:5
(rdbg)
```



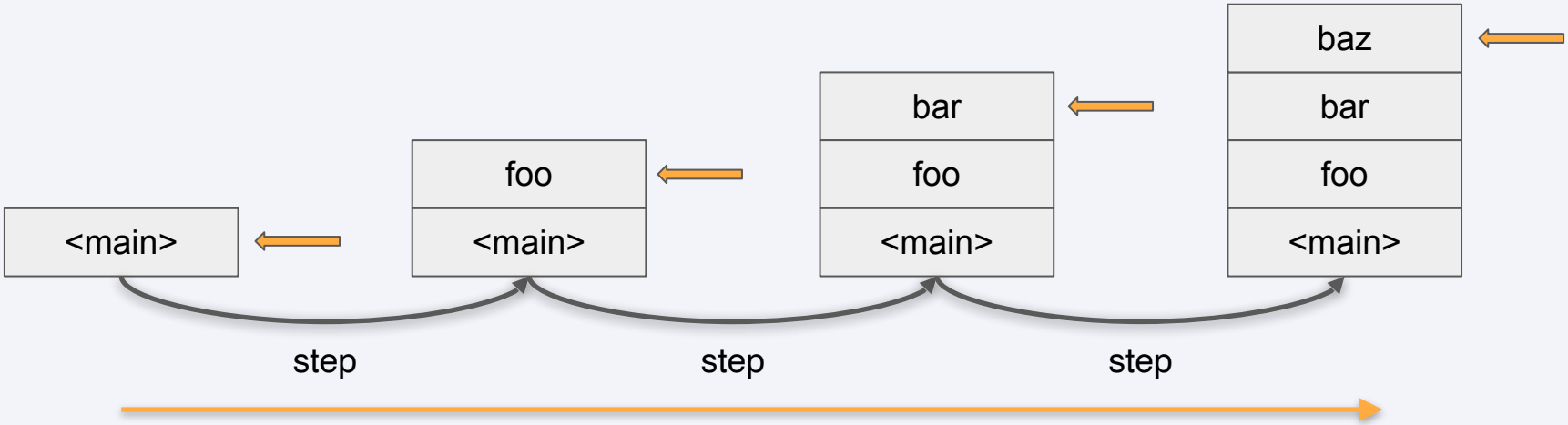
Frame Navigation

```
(rdbg) frame 4    # command
=>   5| result = foo(100)
=>#4    <main> at main.rb:5
(rdbg) list    # command
     1| require "debug"
     2| load "lib.rb"
     3|
     4| binding.b
=>   5| result = foo(100)
     6|
     7| if result == 202
     8|   puts("foo works as expected")
     9| else
    10|   puts("foo returned incorrect value: #{result}")
(rdbg)
```

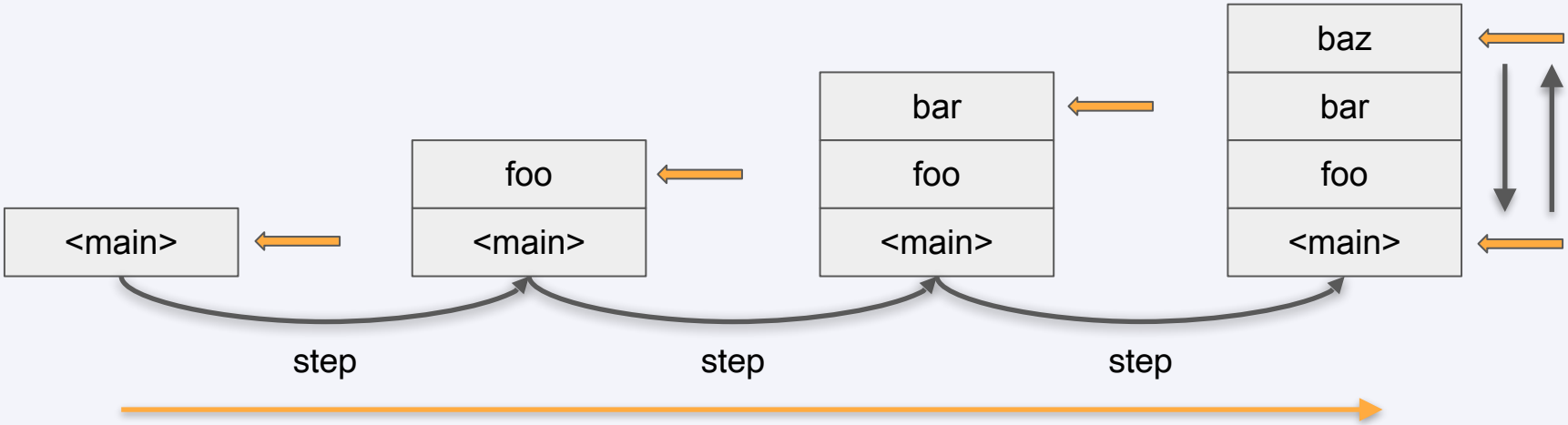


Step-debugging + Frame Navigation

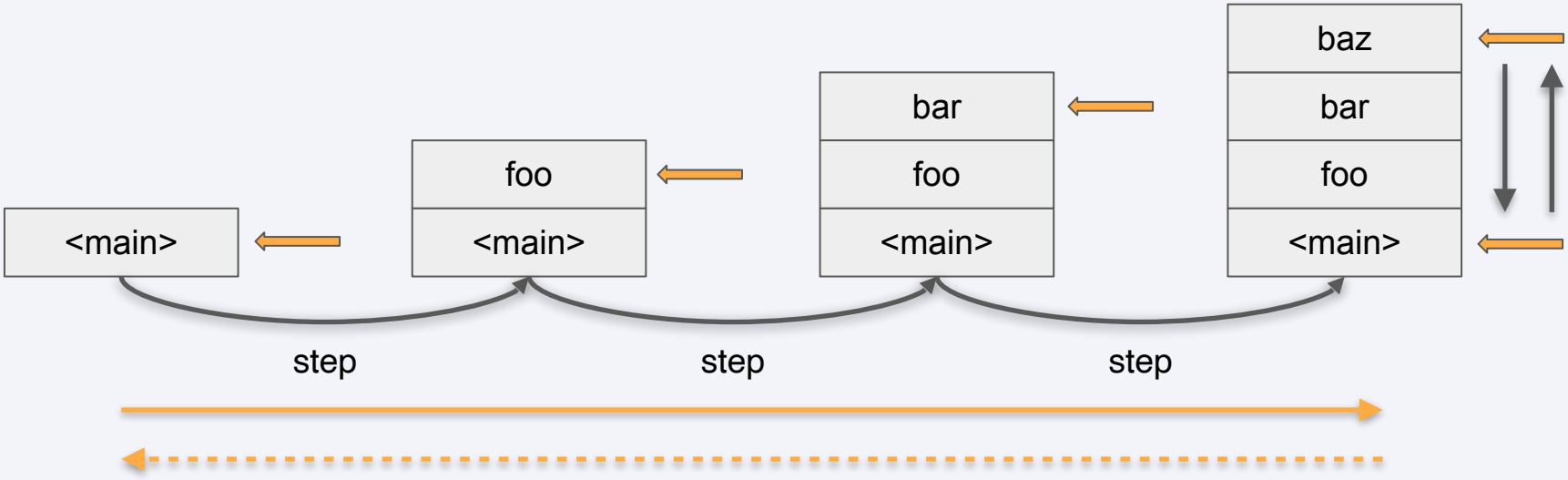
Step-debugging + Frame Navigation



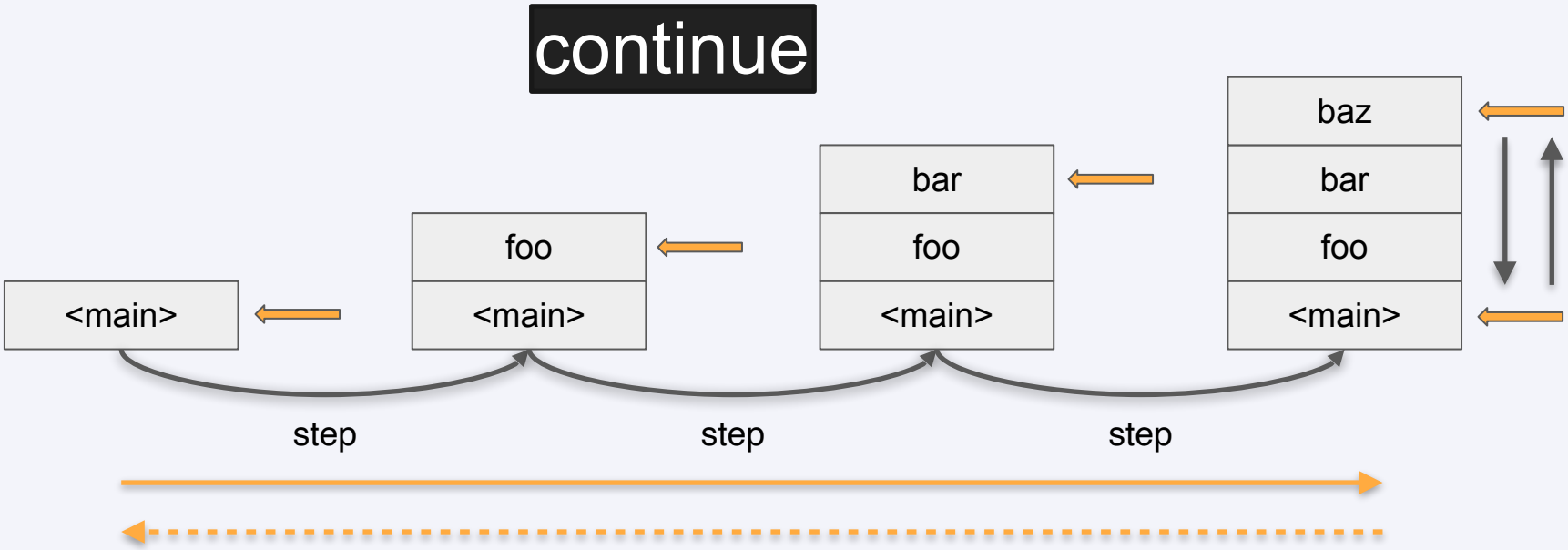
Step-debugging + Frame Navigation



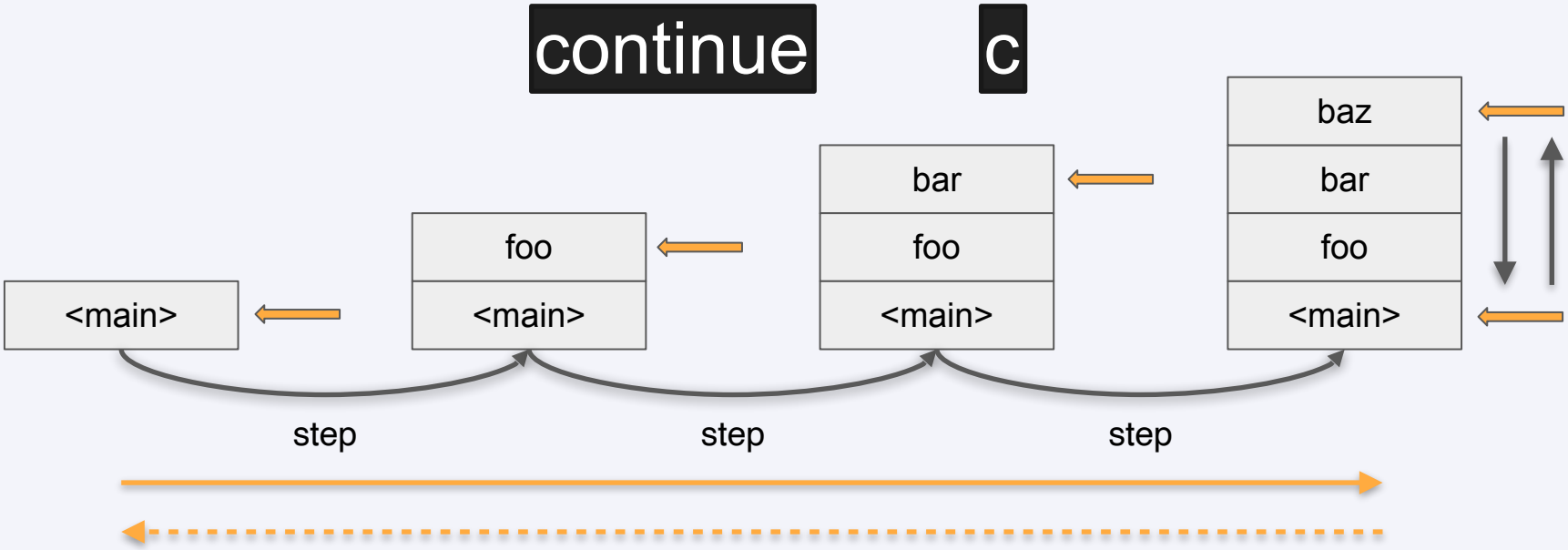
Step-debugging + Frame Navigation



Step-debugging + Frame Navigation



Step-debugging + Frame Navigation



Step-debugging + Frame Navigation - Commands

	Command
Step in	s[tep]
Step over	n[ext]
Finish	fin[ish]
Move to frame	f[rame] <id>
Move up a frame	up
Move down a frame	down
Continue	c[ontinue]

	Command
Show backtrace	bt, backtrace
Display variables	i[nfo]

Breakpoint Commands

Teleporting in our program

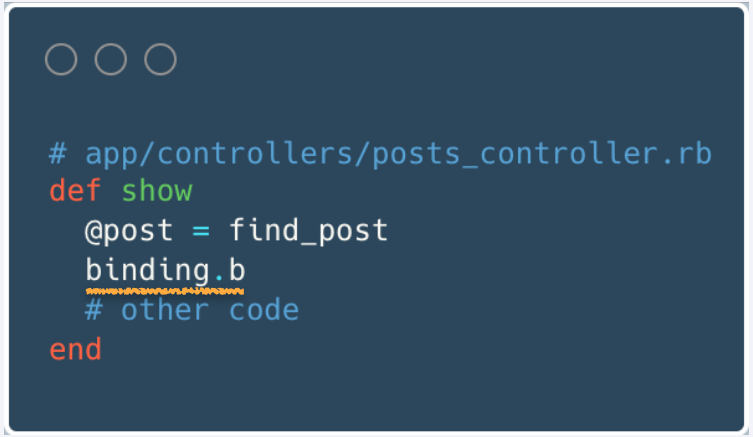
shopify

Breakpoint Commands - Example

```ruby
# app/controllers/posts_controller.rb
def show
  @post = find_post
  binding.b
  # other code
end
```

shopify

Breakpoint Commands - Example



```ruby
# app/controllers/posts_controller.rb
def show
  @post = find_post
  binding.b
  # other code
end
```

shopify

Breakpoint Commands - Example

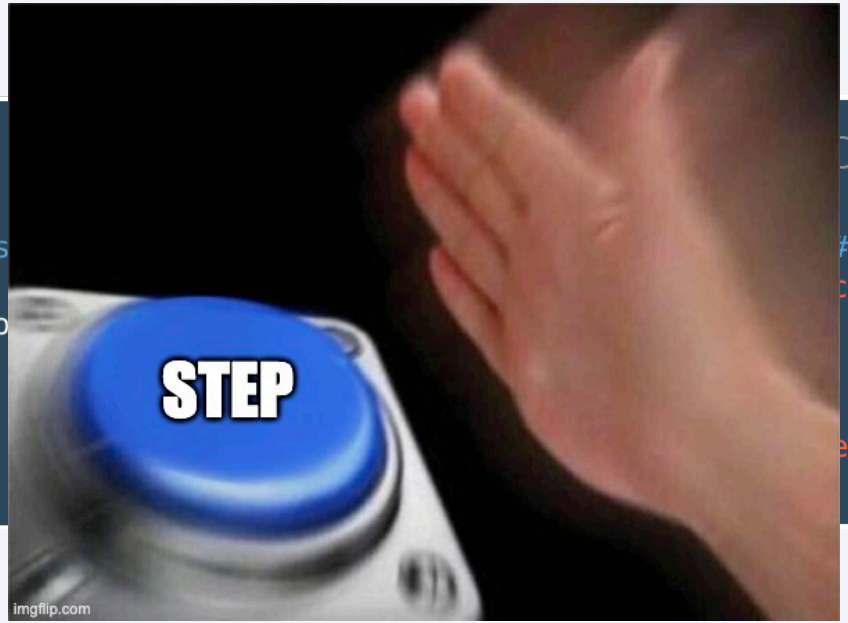
```ruby
# app/controllers/posts_controller.rb
def show
  @post = find_post
  binding.b
  # other code
end
```

```ruby
# app/models/post.rb
class Post
  def title
    # might have a bug here
  end
end
```

shopify

Breakpoint Commands - Don’t Do This: The Stepping Warrior Way

```
# app/controllers
def show
  @post = find_po
  binding.b
  # other code
end
```




```
# app/models/post.rb
class Post
  def title
    # might have a bug here
  end
end
```

shopify

Breakp ...pping Warrior Way

Breakpoint Commands - Don’t Do This: The Pry Way

1. Open app/models/post.rb
2. Put another binding.b in Post#title
3. Restart the program
4. Stop at PostsController#show again
5. Type @post.title

shopify

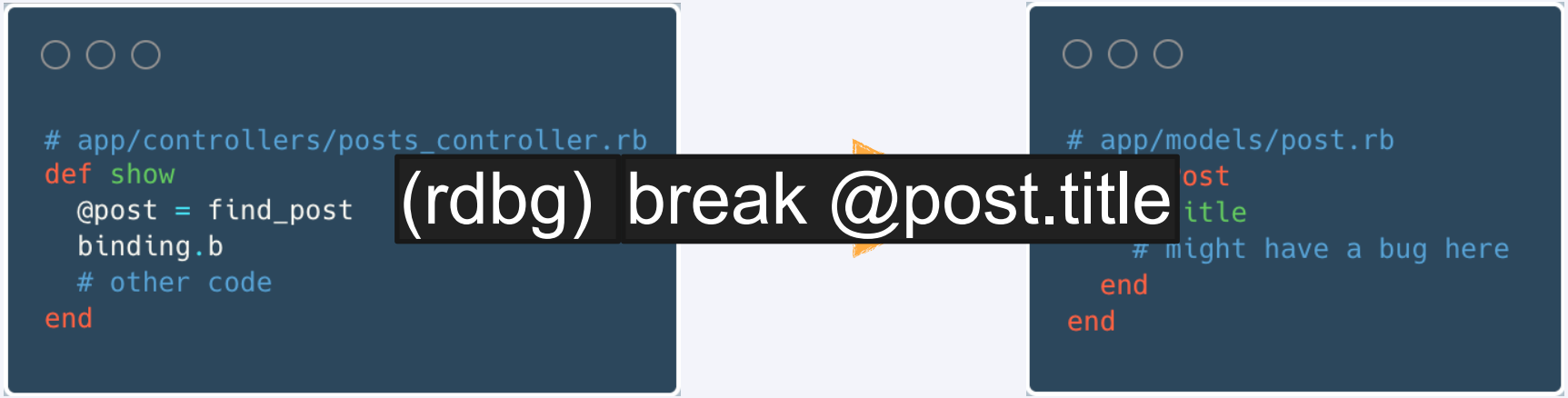
Breakpoint Commands - Teleport with breakpoints

```ruby
# app/controllers/posts_controller.rb
def show
  @post = find_post
  binding.b
  # other code
end
```

```ruby
# app/models/post.rb
class Post
  def title
    # might have a bug here
  end
end
```

shopify

Breakpoint Commands - Teleport with breakpoints



shopify

Breakpoint Commands

	Command
On line	b[reak] <line>
On file:line	b[reak] <file>:<line>
On a method	b[reak] <class>#<method>
On a class method	b[reak] <class>.<method>
On an instance's method	b[reak] <obj>.<method>
On exceptions	catch <exception_class>

	Command
List all breakpoints	b[reak]
Delete a breakpoint	del[ete] <id>
Delete all breakpoints	del[ete]

Breakpoint Commands

	Command
On line	b[reak] <line>
On file:line	b[reak] <file>:<line>
On a method	b[reak] <class>#<method>
On a class method	b[reak] <class>.<method>
On an instance's method	b[reak] <obj>.<method>
On exceptions	catch <exception_class>

	Command
List all breakpoints	b[reak]
Delete a breakpoint	del[ete] <id>
Delete all breakpoints	del[ete]

shopify

Breakpoint Commands

	Command
On line	b[reak] <line>
On file:line	b[reak] <file>:<line>
On a method	b[reak] <class>#<method>
On a class method	b[reak] <class>.<method>
On an instance's method	b[reak] <obj>.<method>
On exceptions	catch <exception_class>

	Command
List all breakpoints	b[reak]
Delete a breakpoint	del[ete] <id>
Delete all breakpoints	del[ete]

Breakpoint Commands

	Command
On line	b[reak] <line>
On file:line	b[reak] <file>:<line>
On a method	b[reak] <class>#<method>
On a class method	b[reak] <class>.<method>
On an instance's method	b[reak] <obj>.<method>
On exceptions	catch <exception_class>

	Command
List all breakpoints	b[reak]
Delete a breakpoint	del[ete] <id>
Delete all breakpoints	del[ete]

Breakpoint Commands

	Command
On line	b[reak] <line>
On file:line	b[reak] <file>:<line>
On a method	b[reak] <class>#<method>
On a class method	b[reak] <class>.<method>
On an instance's method	b[reak] <obj>.<method>
On exceptions	catch <exception_class>

	Command
List all breakpoints	b[reak]
Delete a breakpoint	del[ete] <id>
Delete all breakpoints	del[ete]

Breakpoint Commands

	Command
On line	b[reak] <line>
On file:line	b[reak] <file>:<line>
On a method	b[reak] <class>#<method>
On a class method	b[reak] <class>.<method>
On an instance's method	b[reak] <obj>.<method>
On exceptions	catch <exception_class>

	Command
List all breakpoints	b[reak]
Delete a breakpoint	del[ete] <id>
Delete all breakpoints	del[ete]

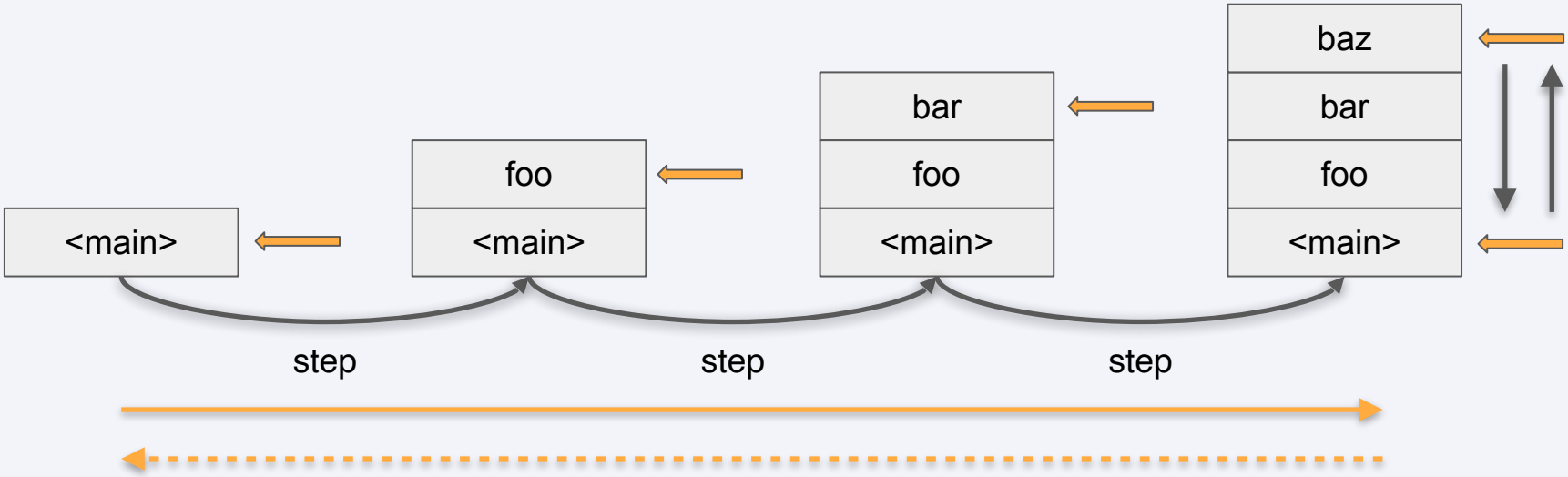
shopify

Breakpoint Commands

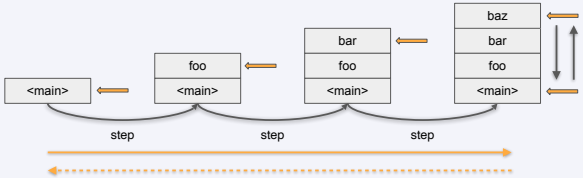
	Command
On line	b[reak] <line>
On file:line	b[reak] <file>:<line>
On a method	b[reak] <class>#<method>
On a class method	b[reak] <class>.<method>
On an instance's method	b[reak] <obj>.<method>
On exceptions	catch <exception_class>

	Command
List all breakpoints	b[reak]
Delete a breakpoint	del[ete] <id>
Delete all breakpoints	del[ete]

Step-debugging + Frame Navigation + Breakpoint Commands

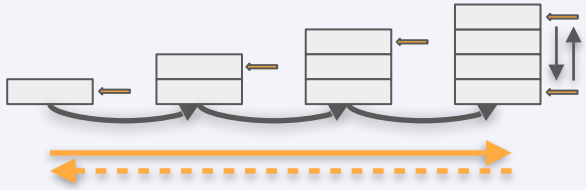


Step-debugging + Frame Navigation + Breakpoint Commands



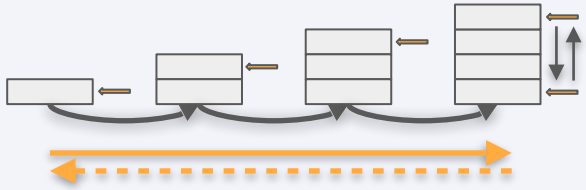
Step-debugging + Frame Navigation + Breakpoint Commands

Investigation



Step-debugging + Frame Navigation + Breakpoint Commands

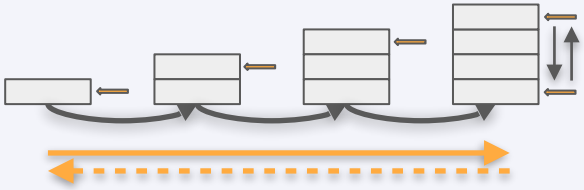
Investigation



Set destination (breakpoint)

Step-debugging + Frame Navigation + Breakpoint Commands

Investigation



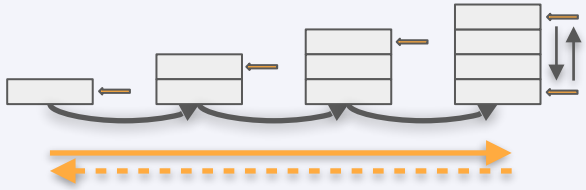
Set destination (breakpoint)

continue

Triggers breakpoint

Step-debugging + Frame Navigation + Breakpoint Commands

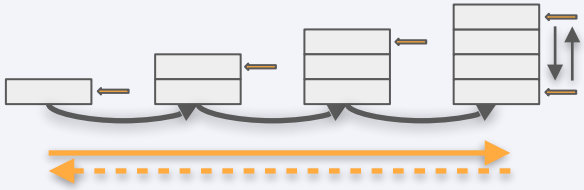
Investigation



Set destination (breakpoint)

continue

Investigation - 2



Scriptable Breakpoint

Debug like we program

shopify

Scriptable Breakpoints

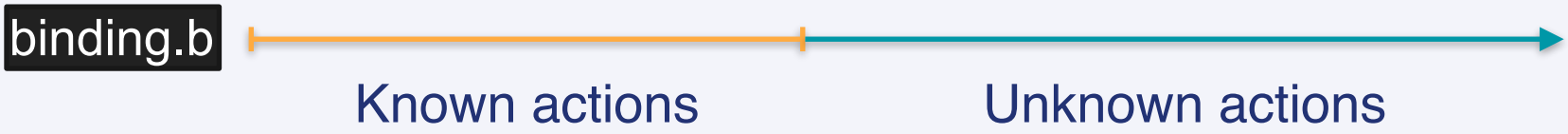
```
binding.b
```

shopify

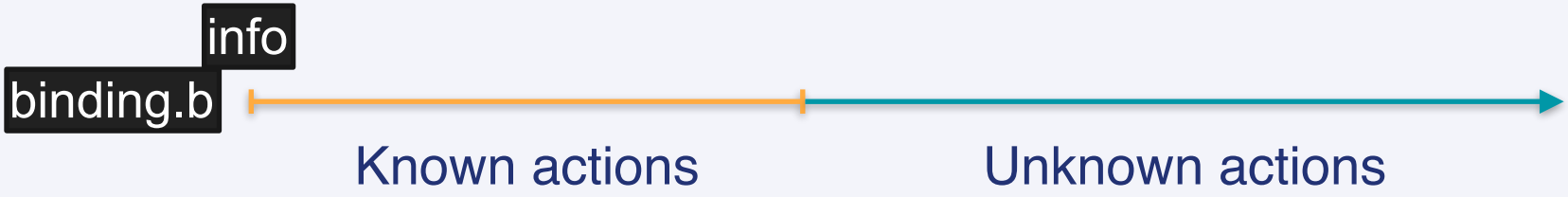
Scriptable Breakpoints

shopify

Scriptable Breakpoints



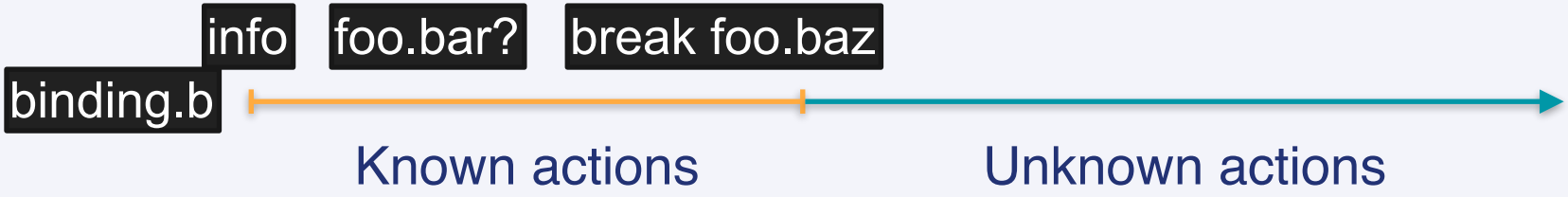
Scriptable Breakpoints



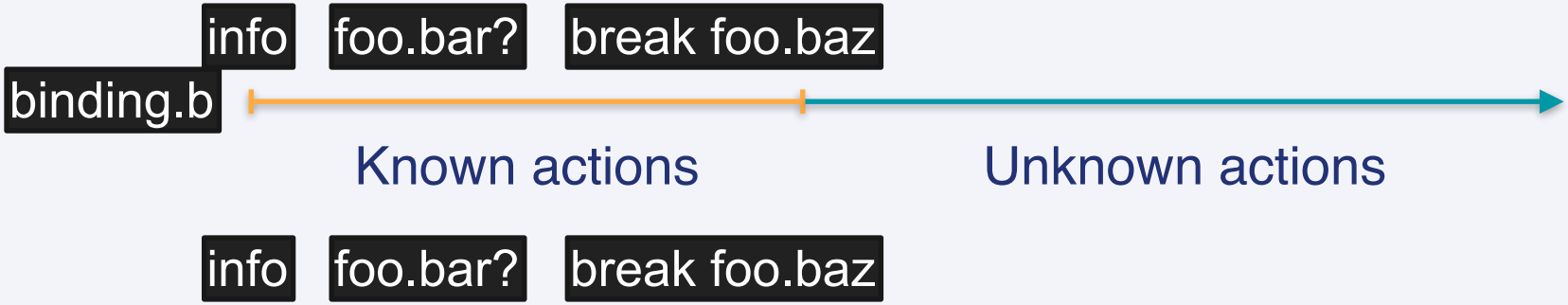
Scriptable Breakpoints

shopify

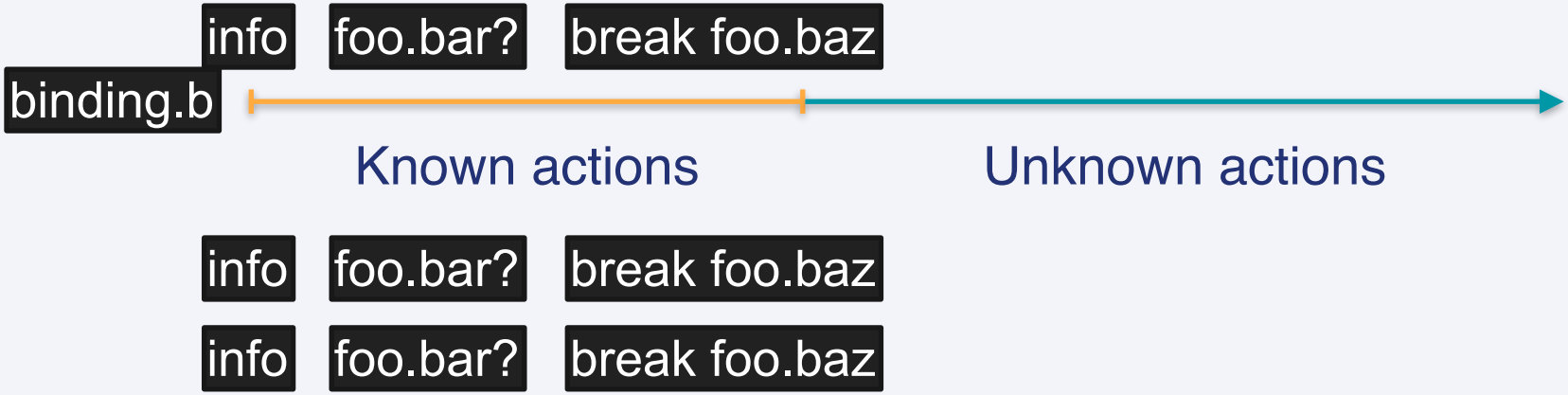
Scriptable Breakpoints



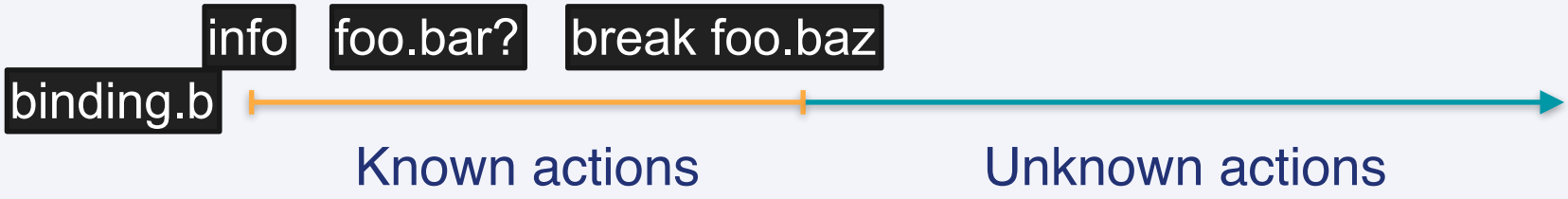
Scriptable Breakpoints



Scriptable Breakpoints



Scriptable Breakpoints



Scriptable Breakpoints

```
binding.b(pre: "info ;; foo.bar? ;; break foo.baz")
```

Known actions

Unknown actions

shopify

Scriptable Breakpoints - The "pre" and "do" keywords

Scriptable Breakpoints - The "pre" and "do" keywords

- binding.b(pre: "cmd")
    1. Execute <cmd>
    2. Stops at the breakpoint

Scriptable Breakpoints - The "pre" and "do" keywords

- binding.b(pre: "cmd")
    1. Execute <cmd>
    2. Stops at the breakpoint
- binding.b(do: "cmd")
    1. Execute <cmd>
    2. Continue the program

shopify

Scriptable Breakpoints - Multiple Commands

- Separate commands or expressions with ";;"
- binding.b(do: "cmd1 ;; cmd2")
- binding.b(pre: "var = foo ;; bar(var)")

shopify

Scriptable Breakpoints - Breakpoint Commands Support

- (rdbg) break Foo#bar do: info
- (rdbg) catch StandardError pre: bt

Scriptable Breakpoints - Example

```ruby
# create an old encryptor
old_secret = SecureRandom.base64(24)
old_encryptor = ActiveSupport::MessageEncryptor.new old_secret

# create a new encryptor
new_secret = SecureRandom.base64(24)
new_encryptor = ActiveSupport::MessageEncryptor.new new_secret

# let the new encryptor rotate to old_secret when needed
new_encryptor.rotate old_secret

# the new encryptor should decrypt the message after rotating to the old secret
msg = new_encryptor.decrypt_and_verify(old_encryptor.encrypt_and_sign(nil))
puts("Message is #{msg.inspect}") #=> Message is nil
```

Scriptable Breakpoints - Example

```ruby
# create an old encryptor
old_secret = SecureRandom.base64(24)
old_encryptor = ActiveSupport::MessageEncryptor.new old_secret

# create a new encryptor
new_secret = SecureRandom.base64(24)
new_encryptor = ActiveSupport::MessageEncryptor.new new_secret

# let the new encryptor rotate to old_secret when needed
new_encryptor.rotate old_secret

# the new encryptor should decrypt the message after rotating to the old secret
msg = new_encryptor.decrypt_and_verify(old_encryptor.encrypt_and_sign(nil))
puts("Message is #{msg.inspect}") #=> Message is nil
```

Scriptable Breakpoints - Example

```ruby
# create an old encryptor
old_secret = SecureRandom.base64(24)
old_encryptor = ActiveSupport::MessageEncryptor.new old_secret

# create a new encryptor
new_secret = SecureRandom.base64(24)
new_encryptor = ActiveSupport::MessageEncryptor.new new_secret

# let the new encryptor rotate to old_secret when needed
new_encryptor.rotate old_secret

# the new encryptor should decrypt the message after rotating to the old secret
msg = new_encryptor.decrypt_and_verify(old_encryptor.encrypt_and_sign(nil))
puts("Message is #{msg.inspect}") #=> Message is nil
```

Scriptable Breakpoints - Example

```ruby
# create an old encryptor
old_secret = SecureRandom.base64(24)
old_encryptor = ActiveSupport::MessageEncryptor.new old_secret

# create a new encryptor
new_secret = SecureRandom.base64(24)
new_encryptor = ActiveSupport::MessageEncryptor.new new_secret

# let the new encryptor rotate to old_secret when needed
new_encryptor.rotate old_secret

# the new encryptor should decrypt the message after rotating to the old secret
msg = new_encryptor.decrypt_and_verify(old_encryptor.encrypt_and_sign(nil))
puts("Message is #{msg.inspect}") #=> Message is nil
```

Scriptable Breakpoints - Example

```
# create an old encryptor
old_secret = SecureRandom.base64(24)
old_encryptor = ActiveSupport::MessageEncryptor.new old_secret
```

```
activesupport-7.0.3.1/lib/active_support/message_verifier.rb:178:in `verify':
ActiveSupport::MessageVerifier::InvalidSignature (ActiveSupport::MessageVerifier::InvalidSignature)
```

```
new_encryptor.rotate old_secret

# the new encryptor should decrypt the message after rotating to the old secret
msg = new_encryptor.decrypt_and_verify(old_encryptor.encrypt_and_sign(nil))
puts("Message is #{msg.inspect}") #=> Message is nil
```

Scriptable Breakpoints - Example

```ruby
# create an old encryptor
old_secret = SecureRandom.base64(24)
old_encryptor = ActiveSupport::MessageEncryptor.new old_secret

# create a new encryptor
new_secret = SecureRandom.base64(24)
new_encryptor = ActiveSupport::MessageEncryptor.new new_secret

# let the new encryptor rotate to old_secret when needed
new_encryptor.rotate old_secret

binding.b(do: "i ;; break new_encryptor.decrypt_and_verify pre: i")

# the new encryptor should decrypt the message after rotating to the old secret
msg = new_encryptor.decrypt_and_verify(old_encryptor.encrypt_and_sign(nil))
puts("Message is #{msg.inspect}") #=> Message is nil
```

Scriptable Breakpoints - Example

```ruby
# create an old encryptor
old_secret = SecureRandom.base64(24)
old_encryptor = ActiveSupport::MessageEncryptor.new old_secret

# create a new encryptor
new_secret = SecureRandom.base64(24)
new_encryptor = ActiveSupport::MessageEncryptor.new new_secret

# le info new encryptor rotate to old_secret when needed
new_encryptor.rotate old_secret

binding.b(do: "i ;; break new_encryptor.decrypt_and_verify pre: i")

# the new encryptor should decrypt the message after rotating to the old secret
msg = new_encryptor.decrypt_and_verify(old_encryptor.encrypt_and_sign(nil))
puts("Message is #{msg.inspect}") #=> Message is nil
```

Scriptable Breakpoints - Example

```ruby
# create an old encryptor
old_secret = SecureRandom.base64(24)
old_encryptor = ActiveSupport::MessageEncryptor.new old_secret

# create a new encryptor
new_secret = SecureRandom.base64(24)
new_encryptor = ActiveSupport::MessageEncryptor.new new_secret

# le info new e break new_encryptor.decrypt_and_verify
new_encryptor.rotate old_secret

binding.b(do: "i ;; break new_encryptor.decrypt_and_verify pre: i")

# the new encryptor should decrypt the message after rotating to the old secret
msg = new_encryptor.decrypt_and_verify(old_encryptor.encrypt_and_sign(nil))
puts("Message is #{msg.inspect}") #=> Message is nil
```

Scriptable Breakpoints - Example

```ruby
# create an old encryptor
old_secret = SecureRandom.base64(24)
old_encryptor = ActiveSupport::MessageEncryptor.new old_secret

# create a new encryptor
new_secret = SecureRandom.base64(24)
new_encryptor = ActiveSupport::MessageEncryptor.new new_secret

# le info new e break new_encryptor.decrypt_and_verify info
new_encryptor.rotate old_secret

binding.b(do: "i ;; break new_encryptor.decrypt_and_verify pre: i")

# the new encryptor should decrypt the message after rotating to the old secret
msg = new_encryptor.decrypt_and_verify(old_encryptor.encrypt_and_sign(nil))
puts("Message is #{msg.inspect}") #=> Message is nil
```

Scriptable Breakpoints - Example

```ruby
# create an old encryptor
old_secret = SecureRandom.base64(24)
old_encryptor = ActiveSupport::MessageEncryptor.new old_secret

# create a new encryptor
new_secret = SecureRandom.base64(24)
new_encryptor = ActiveSupport::MessageEncryptor.new new_secret

# le info new e break new_encryptor.decrypt_and_verify info
new_encryptor.rotate old_secret

binding.b(do: "i ;; break new_encryptor.decrypt_and_verify pre: i")

# the new encryptor should decrypt the message after rotating to the old secret
msg = new_encryptor.decrypt_and_verify(old_encryptor.encrypt_and_sign(nil))
puts("Message is #{msg.inspect}") #=> Message is nil
```

Scriptable Breakpoints - Example

```ruby
# create an old encryptor
old_secret = SecureRandom.base64(24)
old_encryptor = ActiveSupport::MessageEncryptor.new old_secret

# create a new encryptor
new_secret = SecureRandom.base64(24)
new_encryptor = ActiveSupport::MessageEncryptor.new new_secret

# let the new e
new_encryptor.rotate old_secret

binding.b(do: "i ;; break new_encryptor.decrypt_and_verify pre: i")

# the new encryptor should decrypt the message after rotating to the old secret
msg = new_encryptor.decrypt_and_verify(old_encryptor.encrypt_and_sign(nil))
puts("Message is #{msg.inspect}") #=> Message is nil
```

break new_encryptor.decrypt_and_verify

info

Scriptable Breakpoints - Example

```ruby
# create an old encryptor
old_secret = SecureRandom.base64(24)
old_encryptor = ActiveSupport::MessageEncryptor.new old_secret

# create a new encryptor
new_secret = SecureRandom.base64(24)
new_encryptor = ActiveSupport::MessageEncryptor.new new_secret

# let the new encryptor rotate to old_secret when needed
new_encryptor.rotate old_secret

binding.b(do: "i ;; break new_encryptor.decrypt_and_verify pre: i")

# the new encryptor should decrypt the message after rotating to the old secret
msg = new_encryptor.decrypt_and_verify(old_encryptor.encrypt_and_sign(nil))
puts("Message is #{msg.inspect}") #=> Message is nil
```

info

Scriptable Breakpoints - Example

```ruby
# create an old encryptor
old_secret = SecureRandom.base64(24)
old_encryptor = ActiveSupport::MessageEncryptor.new old_secret

# create a new encryptor
new_secret = SecureRandom.base64(24)
new_encryptor = ActiveSupport::MessageEncryptor.new new_secret

# let the new encryptor rotate to old_secret when needed
new_encryptor.rotate old_secret

binding.b(do: "i ;; break new_encryptor.decrypt_and_verify pre: i")

# the new encryptor should decrypt the message after rotating to the old secret
msg = new_encryptor.decrypt_and_verify(old_encryptor.encrypt_and_sign(nil))
puts("Message is #{msg.inspect}") #=> Message is nil
```

```
[12, 21] in test.rb
    12| new_encryptor = ActiveSupport::MessageEncryptor.new new_secret
    13|
    14| # let the new encryptor rotate to old_secret when needed
    15| new_encryptor.rotate old_secret
    16|
=>  17| binding.b(do: "i ;; break new_encryptor.decrypt_and_verify pre: i")
    18|
    19| # the new encryptor should descrypt the message after rotating to the old secret
    20| msg = new_encryptor.decrypt_and_verify(old_encryptor.encrypt_and_sign(nil))
    21| puts("Message is #{msg.inspect}") #=> Message is nil
=>#0    <main> at test.rb:17
(rdbg:binding.break) i
%self = main
old_secret = "2LRxl68OgB/023KjhIkFzHck1pSJ0HvL"
old_encryptor = #<ActiveSupport::MessageEncryptor:0x00000001046be388 @secret="2LRxl68OgB/023KjhIkFzHck1pSJ0HvL", @sign_secret=ni...>
new_secret = "BwIjFx+YSGRbkp9yU219CkLiwVeQJj13"
new_encryptor = #<ActiveSupport::MessageEncryptor:0x00000001046bd348 @secret="BwIjFx+YSGRbkp9yU219CkLiwVeQJj13", @sign_secret=ni...>
msg = nil
(rdbg:binding.break) break new_encryptor.decrypt_and_verify pre: i
#0  BP - Method  new_encryptor.decrypt_and_verify at /Users/hung-wulo/.gem/ruby/3.1.2/gems/activesupport-7.0.3.1/lib/active_support/
messages/rotator.rb:21 pre: i
[17, 26] in ~/.gem/ruby/3.1.2/gems/activesupport-7.0.3.1/lib/active_support/messages/rotator.rb
    17|
    18|       module Encryptor
    19|         include Rotator
    20|
    21|         def decrypt_and_verify(*args, on_rotation: @on_rotation, **options)
=>  22|           super
    23|         rescue MessageEncryptor::InvalidMessage, MessageVerifier::InvalidSignature
    24|           run_rotations(on_rotation) { |encryptor| encryptor.decrypt_and_verify(*args, **options) } || raise
    25|         end
    26|
=>#0    ActiveSupport::Messages::Rotator::Encryptor#decrypt_and_verify(args=["TkxwcjlMWpFalU0SlNYQUlZWk03UT09LS0zQ29..., on_rotatio
n=nil, options={}) at ~/.gem/ruby/3.1.2/gems/activesupport-7.0.3.1/lib/active_support/messages/rotator.rb:22
```

```
[12, 21] in test.rb
    12| new_encryptor = ActiveSupport::MessageEncryptor.new new_secret
    13|
    14| # let the new encryptor rotate to old_secret when needed
    15| new_encryptor.rotate old_secret
    16|
=>  17| binding.b(do: "i ;; break new_encryptor.decrypt_and_verify pre: i")
    18|
    19| # the new encryptor should descrypt the message after rotating to the old secret
    20| msg = new_encryptor.decrypt_and_verify(old_encryptor.encrypt_and_sign(nil))
    21| puts("Message is #{msg.inspect}") #=> Message is nil
=>#0    <main> at test.rb:17
(rdbg:binding.break) i
%self = main
old_secret = "2LRxl68OgB/023KjhIkFzHck1pSJ0HvL"
old_encryptor = #<ActiveSupport::MessageEncryptor:0x00000001046be388 @secret="2LRxl68OgB/023KjhIkFzHck1pSJ0HvL", @sign_secret=ni...>
new_secret = "BwIjFx+YSGRbkp9yU219CkLiwVeQJj13"
new_encryptor = #<ActiveSupport::MessageEncryptor:0x00000001046bd348 @secret="BwIjFx+YSGRbkp9yU219CkLiwVeQJj13", @sign_secret=ni...>
msg = nil
(rdbg:binding.break) break new_encryptor.decrypt_and_verify pre: i
#0  BP - Method  new_encryptor.decrypt_and_verify at /Users/hung-wulo/.gem/ruby/3.1.2/gems/activesupport-7.0.3.1/lib/active_support/
messages/rotator.rb:21 pre: i
[17, 26] in ~/.gem/ruby/3.1.2/gems/activesupport-7.0.3.1/lib/active_support/messages/rotator.rb
    17|
    18|       module Encryptor
    19|         include Rotator
    20|
    21|         def decrypt_and_verify(*args, on_rotation: @on_rotation, **options)
=>  22|           super
    23|         rescue MessageEncryptor::InvalidMessage, MessageVerifier::InvalidSignature
    24|           run_rotations(on_rotation) { |encryptor| encryptor.decrypt_and_verify(*args, **options) } || raise
    25|         end
    26|
=>#0    ActiveSupport::Messages::Rotator::Encryptor#decrypt_and_verify(args=["TkxwcjlMWpFalU0SlNYQUlZWk03UT09LS0zQ29..., on_rotatio
n=nil, options={}) at ~/.gem/ruby/3.1.2/gems/activesupport-7.0.3.1/lib/active_support/messages/rotator.rb:22
```

```
[12, 21] in test.rb
    12| new_encryptor = ActiveSupport::MessageEncryptor.new new_secret
    13|
    14| # let the new encryptor rotate to old_secret when needed
    15| new_encryptor.rotate old_secret
    16|
=>  17| binding.b(do: "i ;; break new_encryptor.decrypt_and_verify pre: i")
    18|
    19| # the new encryptor should descrypt the message after rotating to the old secret
    20| msg = new_encryptor.decrypt_and_verify(old_encryptor.encrypt_and_sign(nil))
    21| puts("Message is #{msg.inspect}") #=> Message is nil
=>#0    <main> at test.rb:17
(rdbg:binding.break) i
%self = main
old_secret = "2LRxl68OgB/023KjhIkFzHck1pSJ0HvL"
old_encryptor = #<ActiveSupport::MessageEncryptor:0x00000001046be388 @secret="2LRxl68OgB/023KjhIkFzHck1pSJ0HvL", @sign_secret=ni...>
new_secret = "BwIjFx+YSGRbkp9yU219CkLiwVeQJj13"
new_encryptor = #<ActiveSupport::MessageEncryptor:0x00000001046bd348 @secret="BwIjFx+YSGRbkp9yU219CkLiwVeQJj13", @sign_secret=ni...>
msg = nil
(rdbg:binding.break) break new_encryptor.decrypt_and_verify pre: i
#0  BP - Method  new_encryptor.decrypt_and_verify at /Users/hung-wulo/.gem/ruby/3.1.2/gems/activesupport-7.0.3.1/lib/active_support/
messages/rotator.rb:21 pre: i
[17, 26] in ~/.gem/ruby/3.1.2/gems/activesupport-7.0.3.1/lib/active_support/messages/rotator.rb
    17|
    18|       module Encryptor
    19|         include Rotator
    20|
    21|         def decrypt_and_verify(*args, on_rotation: @on_rotation, **options)
=>  22|           super
    23|         rescue MessageEncryptor::InvalidMessage, MessageVerifier::InvalidSignature
    24|           run_rotations(on_rotation) { |encryptor| encryptor.decrypt_and_verify(*args, **options) } || raise
    25|         end
    26|
=>#0    ActiveSupport::Messages::Rotator::Encryptor#decrypt_and_verify(args=["TkxwcjlMWpFalU0SlNYQUlZWk03UT09LS0zQ29..., on_rotatio
n=nil, options={}) at ~/.gem/ruby/3.1.2/gems/activesupport-7.0.3.1/lib/active_support/messages/rotator.rb:22
```

```
[17, 26] in ~/.gem/ruby/3.1.2/gems/activesupport-7.0.3.1/lib/active_support/messages/rotator.rb
    17|
    18|       module Encryptor
    19|         include Rotator
    20|
    21|         def decrypt_and_verify(*args, on_rotation: @on_rotation, **options)
=>  22|           super
    23|         rescue MessageEncryptor::InvalidMessage, MessageVerifier::InvalidSignature
    24|           run_rotations(on_rotation) { |encryptor| encryptor.decrypt_and_verify(*args, **options) } || raise
    25|         end
    26|
=>#0    ActiveSupport::Messages::Rotator::Encryptor#decrypt_and_verify(args=["TkxwcjJ2MWpFalU0SlNYQUlZWk03UT09LS0zQ29..., on_rotatio
n=nil, options={}) at ~/.gem/ruby/3.1.2/gems/activesupport-7.0.3.1/lib/active_support/messages/rotator.rb:22
  #1    <main> at test.rb:20

Stop by #0  BP - Method  new_encryptor.decrypt_and_verify at /Users/hung-wulo/.gem/ruby/3.1.2/gems/activesupport-7.0.3.1/lib/active_
support/messages/rotator.rb:21 pre: i
(rdbg:break) i
%self = #<ActiveSupport::MessageEncryptor:0x00000001046bd348 @secret="BwIjFx+YSGRbkp9yU219CkLiwVeQJj13", @sign_secret=nil, @ciph...>
args = ["TkxwcjJ2MWpFalU0SlNYQUlZWk03UT09LS0zQ29xRWJEZGhySFp6QXVyUGxyYVh3PT0=--8fe73db0fa8ed2ac7d3b53e1980b035290a90ba5"]
on_rotation = nil
options = {}
@aead_mode = false
@cipher = "aes-256-cbc"
@digest = "SHA1"
@on_rotation = nil
@options = {}
@rotations = [#<ActiveSupport::MessageEncryptor:0x00000001046bcc18 @secret="2LRxl68OgB/023KjhIkFzHck1pSJ0HvL", @sign_secret=nil, ...
@secret = "BwIjFx+YSGRbkp9yU219CkLiwVeQJj13"
@serializer = Marshal
@sign_secret = nil
@verifier = #<ActiveSupport::MessageVerifier:0x00000001046bd140 @secret="BwIjFx+YSGRbkp9yU219CkLiwVeQJj13", @digest="SHA1", @ser...>
(rdbg)
```

```
[17, 26] in ~/.gem/ruby/3.1.2/gems/activesupport-7.0.3.1/lib/active_support/messages/rotator.rb
    17|
    18|       module Encryptor
    19|         include Rotator
    20|
    21|         def decrypt_and_verify(*args, on_rotation: @on_rotation, **options)
=>  22|           super
    23|         rescue MessageEncryptor::InvalidMessage, MessageVerifier::InvalidSignature
    24|           run_rotations(on_rotation) { |encryptor| encryptor.decrypt_and_verify(*args, **options) } || raise
    25|         end
    26|
=>#0    ActiveSupport::Messages::Rotator::Encryptor#decrypt_and_verify(args=["TkxwcjJ2MWpFalU0SlNYQUlZWk03UT09LS0zQ29..., on_rotation=nil, options={}) at ~/.gem/ruby/3.1.2/gems/activesupport-7.0.3.1/lib/active_support/messages/rotator.rb:22
  #1    <main> at test.rb:20

Stop by #0  BP - Method  new_encryptor.decrypt_and_verify at /Users/hung-wulo/.gem/ruby/3.1.2/gems/activesupport-7.0.3.1/lib/active_support/messages/rotator.rb:21 pre: i
```

```
(rdbg:break) i
%self = #<ActiveSupport::MessageEncryptor:0x00000001046bd348 @secret="BwIjFx+YSGRbkp9yU219CkLiwVeQJj13", @sign_secret=nil, @ciph...>
args = ["TkxwcjJ2MWpFalU0SlNYQUlZWk03UT09LS0zQ29xRWJEZGhySFp6QXVyUGxyYVh3PT0=--8fe73db0fa8ed2ac7d3b53e1980b035290a90ba5"]
on_rotation = nil
options = {}
@aead_mode = false
@cipher = "aes-256-cbc"
@digest = "SHA1"
@on_rotation = nil
@options = {}
@rotations = [#<ActiveSupport::MessageEncryptor:0x00000001046bcc18 @secret="2LRxl68OgB/023KjhIkFzHck1pSJ0HvL", @sign_secret=nil, ...
@secret = "BwIjFx+YSGRbkp9yU219CkLiwVeQJj13"
@serializer = Marshal
@sign_secret = nil
@verifier = #<ActiveSupport::MessageVerifier:0x00000001046bd140 @secret="BwIjFx+YSGRbkp9yU219CkLiwVeQJj13", @digest="SHA1", @ser...>
(rdbg)
```

```
[17, 26] in ~/.gem/ruby/3.1.2/gems/activesupport-7.0.3.1/lib/active_support/messages/rotator.rb
    17|
    18|       module Encryptor
    19|         include Rotator
    20|
    21|         def decrypt_and_verify(*args, on_rotation: @on_rotation, **options)
=>  22|           super
    23|         rescue MessageEncryptor::InvalidMessage, MessageVerifier::InvalidSignature
    24|           run_rotations(on_rotation) { |encryptor| encryptor.decrypt_and_verify(*args, **options) } || raise
    25|         end
    26|
=>#0    ActiveSupport::Messages::Rotator::Encryptor#decrypt_and_verify(args=["TkxwcjJ2MWpFalU0SlNYQUlZWk03UT09LS0zQ29..., on_rotation=nil, options={}) at ~/.gem/ruby/3.1.2/gems/activesupport-7.0.3.1/lib/active_support/messages/rotator.rb:22
  #1    <main> at test.rb:20

Stop by #0  BP - Method  new_encryptor.decrypt_and_verify at /Users/hung-wulo/.gem/ruby/3.1.2/gems/activesupport-7.0.3.1/lib/active_support/messages/rotator.rb:21 pre: i
(rdbg:break) i
%self = #<ActiveSupport::MessageEncryptor:0x00000001046bd348 @secret="BwIjFx+YSGRbkp9yU219CkLiwVeQJj13", @sign_secret=nil, @ciph...>
args = ["TkxwcjJ2MWpFalU0SlNYQUlZWk03UT09LS0zQ29xRWJEZGhySFp6QXVyUGxyYVh3PT0=--8fe73db0fa8ed2ac7d3b53e1980b035290a90ba5"]
on_rotation = nil
options = {}
@aead_mode = false
@cipher = "aes-256-cbc"
@digest = "SHA1"
@on_rotation = nil
@options = {}
@rotations = [#<ActiveSupport::MessageEncryptor:0x00000001046bcc18 @secret="2LRxl68OgB/023KjhIkFzHck1pSJ0HvL", @sign_secret=nil, ...
@secret = "BwIjFx+YSGRbkp9yU219CkLiwVeQJj13"
@serializer = Marshal
@sign_secret = nil
@verifier = #<ActiveSupport::MessageVerifier:0x00000001046bd140 @secret="BwIjFx+YSGRbkp9yU219CkLiwVeQJj13", @digest="SHA1", @ser...>
(rdbg)
```

```
[17, 26] in ~/.gem/ruby/3.1.2/gems/activesupport-7.0.3.1/lib/active_support/messages/rotator.rb
    17|
    18|       module Encryptor
    19|         include Rotator
    20|
    21|         def decrypt_and_verify(*args, on_rotation: @on_rotation, **options)
=>  22|           super
    23|         rescue MessageEncryptor::InvalidMessage, MessageVerifier::InvalidSignature
    24|           run_rotations(on_rotation) { |encryptor| encryptor.decrypt_and_verify(*args, **options) } || raise
    25|         end
    26|
=>#0    ActiveSupport::Messages::Rotator::Encryptor#decrypt_and_verify(args=["TkxwcjJ2MWpFalU0SlNYQUlZWk03UT09LS0zQ29..., on_rotatio
n=nil, options={}) at ~/.gem/ruby/3.1.2/gems/activesupport-7.0.3.1/lib/active_support/messages/rotator.rb:22
  #1    <main> at test.rb:20

Stop by #0  BP - Method  new_encryptor.decrypt_and_verify at /Users/hung-wulo/.gem/ruby/3.1.2/gems/activesupport-7.0.3.1/lib/active_
support/messages/rotator.rb:21 pre: i
(rdbg:break) i
%self = #<ActiveSupport::MessageEncryptor:0x00000001046bd348 @secret="BwIjFx+YSGRbkp9yU219CkLiwVeQJj13", @sign_secret=nil, @ciph...>
args = ["TkxwcjJ2MWpFalU0SlNYQUlZWk03UT09LS0zQ29xRWJEZGhySFp6QXVyUGxyYVh3PT0=--8fe73db0fa8ed2ac7d3b53e1980b035290a90ba5"]
on_rotation = nil
options = {}
@aead_mode = false
@cipher = "aes-256-cbc"
@digest = "SHA1"
@on_rotation = nil
@options = {}
@rotations = [#<ActiveSupport::MessageEncryptor:0x00000001046bcc18 @secret="2LRxl68OgB/023KjhIkFzHck1pSJ0HvL", @sign_secret=nil, ...
@secret = "BwIjFx+YSGRbkp9yU219CkLiwVeQJj13"
@serializer = Marshal
@sign_secret = nil
@verifier = #<ActiveSupport::MessageVerifier:0x00000001046bd140 @secret="BwIjFx+YSGRbkp9yU219CkLiwVeQJj13", @digest="SHA1", @ser...>
(rdbg)
```

Scriptable Breakpoints - Benefits

- Reduces manual operations and human errors
- Helps you plan debugging workflow ahead
- Utilises editor features like autocompletion
- Makes our debugging experience sharable

Why ruby/debug

Advantage of ruby/debug
• Maintained by Ruby core • Colorised output • Powerful breakpoints and tracers • Advanced remote debugging support • Native VSCode integration

Why ruby/debug

Advantage of ruby/debug	Disadvantage of ruby/debug
• Maintained by Ruby core • Colorised output • Powerful breakpoints and tracers • Advanced remote debugging support • Native VSCode integration	• Doesn’t work with Fiber • Less flexible thread control • Less learning resouces

Why ruby/debug

Advantage of ruby/debug	Disadvantage of ruby/debug
• Maintained by Ruby core • Colorised output • Powerful breakpoints and tracers • Advanced remote debugging support • Native VSCode integration	• Doesn't work with Fiber • Less flexible thread control • arning resouces

https://st0012.dev/from-byebug-to-ruby-debug

Wrapping Up

- Step-debugging
- Frame Navigation

shopify

Wrapping Up

- **Step-debugging**
- **Frame Navigation**

Detailed investigation

shopify

Wrapping Up

- **Step-debugging**
- **Frame Navigation**
- **Breakpoint Commands**

Detailed investigation

Keeps the flow uninterrupted

Wrapping Up

- **Step-debugging**
- **Frame Navigation**
- **Breakpoint Commands**
- **Scriptable Breakpoint**

Detailed investigation

Keeps the flow uninterrupted

Program our debugging session

shopify

Happy debugging!

shopify