## **Fiber**

This lesson introduces the concept of Fibers in Ruby, which is, in a sense, the equivalent of coroutines.

## **Fibers**

In this lesson, we'll learn the basics of working with fibers.

### **Creating**

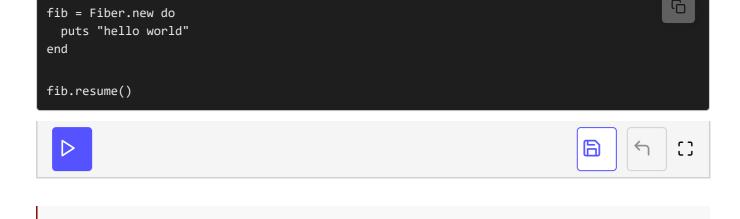
It is trivial to create a fiber. For example:

```
fib = Fiber.new do
end
```

Once created, we can run a fiber using the **resume()** instance method on a fiber object. e.g.

```
fib = Fiber.new do end fib.resume()
```

The above fiber doesn't do anything useful. We can print a "hello world" message in the fiber's code block as follows:



Or more succinctly as:

#### **Current Fiber**

We can get the current fiber using Fiber.current similar to how we get the current thread using Thread.current. An example is shown below:

```
require 'fiber'

# prints the id fo the puts Fiber.current.__id__

\[ \begin{align*}
\text{ } & \lefta & \text{ } \left
```

Note that each thread comes with its root fiber as demonstrated below:

```
require 'fiber'

fib = Fiber.current

Throad now do
```

# **Query Status**

We can query the status of a fiber as follows:

```
require 'fiber'

fib = Fiber.new do
  puts "hello"
end

# prints true
puts fib.alive?
fib.resume
#prints false
puts fib.alive?
```

Attempting to resume a fiber that has already run its course, results in an error as demonstrated below:

```
fib = Fiber.new do
  puts "hello"
end

fib.resume
# A second resume results in an error
fib.resume
```





