The Ruby **Each** Loop

The Ruby method each allows you to go over a list of items, without having to keep track of the number of iterations, or having to increase some kind of counter.

It's the Ruby way of doing "repeat until done".

Before you can use **`each`**, you need a collection of items like an array, a range or a hash.

For example:

```
numbers = [1, 3, 5, 7]
```

Then you can use each like this:

```
numbers.each { |n| puts n }
```

In plain English this is saying:

"For each element in `numbers` print its value."

You tell the each method what to do with every item by using a block.

In this example, the whole thing after `each` is a block:

```
{ |n| puts n }
```

What happens is that **`each`** will use the **block** once for every element in the array & pass every individual element into it, so this `n` is a variable that changes.

Remember:

The purpose of a loop is to iterate or visit ALL the elements from a list, this list can take many forms, but usually it's an array.

There are different ways to do this depending on the situation.

The most common is **using the each method** because you don't need to keep track of the current position within the list.

Ruby does the hard work for you & gives you the individual elements as the `n` variable, but it could be any other valid variable name that you choose.

The Ruby **Each** Loop

Each Method With a Hash

If you want to use **`each`** with a hash you'll need two parameters, one for the **`key`** & another for the **`value`**.

The rest of the syntax is the same & you still need a **block**.

Example:

```
hash = { bacon: 300, coconut: 200 }
```

```
hash.each { |key,value| puts "#{key} price is #{value}" }
```

How to Use Each With Index

If you want to use **`each`** but you need the index number.

You can use the each_with_index method:

animals = ["cat", "dog", "tiger"]

```
animals.each_with_index { |animal, idx| puts "We have a #{animal} with index #{idx}" }
```

This allows you to loop through an array, while having access to the current index.

Remember that the index starts at 0.

Range Looping

You may have noticed that when using the **`times` method** it starts counting from 0.

This can be a bit inconvenient **if you want to start with a different number**.

You can use a **range** & the **`each`** method to have more control over the starting & ending numbers.

Example:

```
(1..10).each { |i| puts i }
```

This will print all the numbers from 1 to 10.