In this lecture, we will discuss…

- ✧ Arrays
  - How they are **created**
  - How to modify **arrays**
  - **Accessing elements** inside arrays

Arrays

- Collection of **object references** (auto-expandable)
- Indexed using `[]` operator (method)
- Can be indexed with **negative numbers** or **ranges**
- **Heterogeneous types allowed** in the same array
- Can use `%w{str1 str2}` for string array creation

Arrays

```ruby
het_arr = [1, "two", :three] # heterogeneous types
puts het_arr[1] # => two (array indices start at 0)

arr_words = %w{ what a great day today! }
puts arr_words[-2] # => day
puts "#{arr_words.first} - #{arr_words.last}" # => what - today!
p arr_words[-3, 2] # => ["great", "day"] (go back 3 and take 2)

# (Range type covered later...)
p arr_words[2..4] # => ["great", "day", "today"]

# Make a String out of array elements separated by ‘,’
puts arr_words.join(',') # => what,a,great,day,today!
```

Arrays

- ✧ Modifying arrays
  - Append: `push` or `<<`
  - Remove: `pop` or `shift`
  - Set: `[]=` (method)

Arrays

- Randomly pull element(s) out with `sample`
- Sort or reverse with `sort!` and `reverse!`

Arrays

```ruby
# You want a stack (LIFO)? Sure
stack = []; stack << "one"; stack.push ("two")
puts stack.pop # => two

# You need a queue (FIFO)? We have those too...
queue = []; queue.push "one"; queue.push "two"
puts queue.shift # => one

a = [5,3,4,2].sort!.reverse!
p a # => [5,4,3,2] (actually modifies the array)
p a.sample(2) # => 2 random elements

a[6] = 33
p a # => [5, 4, 3, 2, nil, nil, 33]
```

Arrays

- Lots of useful array methods
  - `each` – loop through array
  - `select` – filter array by selecting
  - `reject` – filter array by rejecting
  - `map` – modify each element in the array

Many, many others…

Another Important API

http://ruby-doc.org/core-2.2.0/Array.html

Array Processing

```ruby
a = [1, 3, 4, 7, 8, 10]
a.each { |num| print num } # => 1347810 (no new line)
puts # => (print new line)

new_arr = a.select { |num| num > 4 }
p new_arr # => [7, 8, 10]
new_arr = a.select { |num| num < 10 }
           .reject{ |num| num.even? }
p new_arr # => [1, 3, 7]

# Multiply each element by 3 producing new array
new_arr = a.map {|x| x * 3}
p new_arr # => [3, 9, 12, 21, 24, 30]
```

Summary

- ✧ Arrays API is very **flexible** and **powerful**
- ✧ Lots of ways to **process elements** inside the array

What's next?

- ✧ Ranges