



Nested Data Structures

Arrays and Hashes

Lecture Topics

- Examples of multi-level data structs
- Accessing values through method chaining
- Creating a desired structure
- Dealing with ambiguity in complex structures

Why you care.

- Big Data is a HUGE part of Web Development
- Consuming Internet is consuming nested data
- Ability to navigate them easily is a crucial prerequisite of API integration
- Basic Twitter Response

Everything is an Object

- Everything in Ruby is an Object
- Arrays & Hashes are just Object containers
- So you can fill them however you like

```
object_array = [true, "string", 1024, ahash: {topher: "awesome"}]
```

```
object_hash = { bool: true, string: "strings", array: [1,2,3], integer: 88 }
```

Example of a Nested Array

- Grids with rows and columns

```
row_1 = [ "-", "-", "X"]  
row_2 = [ "-", "0", "X"]  
row_3 = [ "-", "-", "0"]
```

```
tic_tac_toe = [ [ "-", "-", "X"],  
                 [ "-", "0", "X"],  
                 [ "-", "-", "0"] ]
```

Example of a Nested Hash

- Hierarchy with named attributes

```
freda = { age: 27 }  
fred  = { age: 25 }
```

Example of a Nested Hash

- Hierarchy with named attributes

```
foxes = { freda: { age: 27 },  
          fred:  { age: 25 } }
```

Example of a Nested Hash

- Hierarchy with named attributes

```
cohorts = { foxes: { freda: { age: 27 },  
                  fred:  { age: 25 } },  
            otters: { olivia: { age: 24 },  
                     oliver: { age: 29 } } }
```

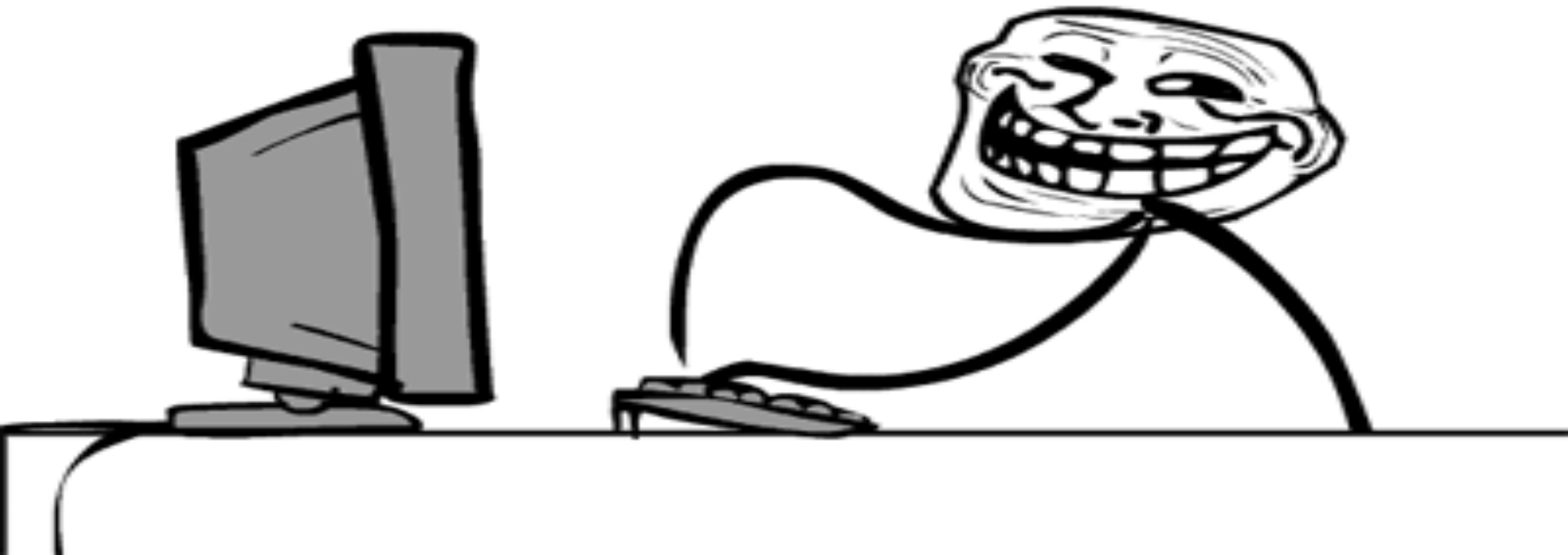

Mixing Arrays and Hashes

- Anything object can be inside the collections

```
cohorts = {  
  foxes: {  
    students: [ { name: "freda", age: 27 },  
                 { name: "fred", age: 25 } ]  
  },  
  otters: {  
    students: [ { name: "olivia", age: 24 },  
                 { name: "oliver", age: 29 } ]  
  }  
}
```

Accessing Values

- How do we work with BIG complicated structures?



Answer: Method Chaining

```
“WARD”.downcase.reverse.capitalize  
# => “Draw”
```

Method Chaining

```
“WARD”.downcase.reverse.capitalize
```

```
“WARD”.downcase    # => “ward”
```

Method Chaining

```
“WARD”.downcase.reverse.capitalize
```

```
“WARD”.downcase    # => “ward”
```

```
“ward”.reverse     # => “draw”
```

Method Chaining

```
“WARD”.downcase.reverse.capitalize
```

```
“WARD”.downcase    # => “ward”
```

```
“ward”.reverse     # => “draw”
```

```
“draw”.capitalize  # => “Draw”
```

Access Values in a Nested Array

- Find an element in tic-tac-toe board

Access Values in a Nested Array

```
tic_tac_toe =  
[ [ "-", "-", "X"],  
  [ "-", "0", "X"],  
  [ "-", "-", "0"]]
```

```
tic_tac_toe.at(1).at(2)
```


Access Values in a Nested Array

```
tic_tac_toe =  
[ [ "-", "-", "X"],  
  [ "-", "0", "X"],  
  [ "-", "-", "0"]]
```

```
tic_tac_toe.at(1).at(2)  
#=> "X"
```

Access Values in a Nested Array

```
tic_tac_toe =  
[ [ "-", "-", "X"],  
  [ "-", "0", "X"],  
  [ "-", "-", "0"]]
```

```
tic_tac_toe.at(1).at(2)  
#=> "X"
```

Access Values in a Nested Array

```
tic_tac_toe =  
[ [ "-", "-", "X"],  
  [ "-", "0", "X"],  
  [ "-", "-", "0"]]
```

```
tic_tac_toe.at(1).at(2)  
#=> "X"
```

Access Values in a Nested Array

```
tic_tac_toe =  
[ [ "-", "-", "X"],  
  [ "-", "0", "X"],  
  [ "-", "-", "0"]]
```

```
tic_tac_toe[1][2]  
#=> "X"
```

Access Values in a Nested Array

```
tic_tac_toe =  
[ [ "-", "-", "X"],  
  [ "-", "0", "X"],  
  [ "-", "-", "0"]]
```

```
tic_tac_toe[1][2]  
#=> "X"
```

Access Values in a Nested Array

```
tic_tac_toe =  
[ [ "-", "-", "X"],  
  [ "-", "0", "X"],  
  [ "-", "-", "0"]]
```

```
tic_tac_toe[1][2]  
#=> "X"
```

Access Values in a Nested Hash

- The age of a student in a cohort

```
cohorts = { foxes: { freda: { age: 27 },  
                  fred:  { age: 25 } },  
            otters: { olivia: { age: 24 },  
                    oliver: { age: 29 } } }
```

Access Values in a Nested Hash

```
cohorts = {  
  foxes: { freda: { age: 27 },  
          fred:  { age: 25 } },  
  otters: { olivia: { age: 24 },  
           oliver: { age: 29 } }  
}
```

```
cohorts.fetch(:foxes).fetch(:freda).fetch(:age)  
#=> 27
```


Access Values in a Nested Hash

```
cohorts = {  
  foxes: { freda: { age: 27 },  
          fred:  { age: 25 } },  
  otters: { olivia: { age: 24 },  
           oliver: { age: 29 } }  
}
```

```
cohorts.fetch(:foxes).fetch(:freda).fetch(:age)  
#=> 27
```

Access Values in a Nested Hash

```
cohorts = {  
  foxes: { freda: { age: 27 },  
          fred:  { age: 25 } },  
  otters: { olivia: { age: 24 },  
            oliver: { age: 29 } }  
}
```

```
cohorts.fetch(:foxes).fetch(:freda).fetch(:age)  
#=> 27
```

Access Values in a Nested Hash

```
cohorts = {  
  foxes: { freda: { age: 27 },  
          fred:  { age: 25 } },  
  otters: { olivia: { age: 24 },  
            oliver: { age: 29 } }  
}
```

```
cohorts.fetch(:foxes).fetch(:freda).fetch(:age)  
#=> 27
```

Access Values in a Nested Hash

```
cohorts = {  
  foxes: { freda: { age: 27 },  
          fred:  { age: 25 } },  
  otters: { olivia: { age: 24 },  
           oliver: { age: 29 } }  
}
```

```
cohorts[:foxes][:freda][:age]  
#=> 27
```

Access Values in a Nested Hash

```
cohorts = {  
  foxes: { freda: { age: 27 },  
          fred:  { age: 25 } },  
  otters: { olivia: { age: 24 },  
           oliver: { age: 29 } }  
}
```

```
cohorts[:foxes][:freda][:age]  
#=> 27
```

Access Values in a Nested Hash

```
cohorts = {  
  foxes: { freda: { age: 27 },  
          fred:  { age: 25 } },  
  otters: { olivia: { age: 24 },  
           oliver: { age: 29 } }  
}
```

```
cohorts[:foxes][:freda][:age]  
#=> 27
```

Access Values in a Nested Hash

```
cohorts = {  
  foxes: { freda: { age: 27 },  
          fred:  { age: 25 } },  
  otters: { olivia: { age: 24 },  
            oliver: { age: 29 } }  
}
```

```
cohorts[:foxes][:freda][:age]  
#=> 27
```

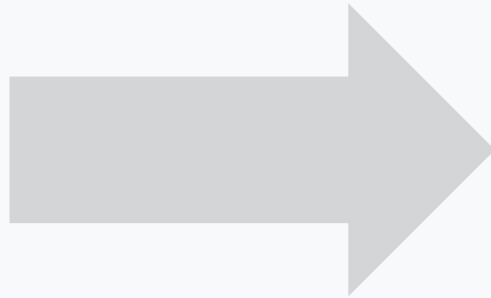
Creating the Structure

- How to populate a desired structure with the right values?

3x3 array with Sample values from a 3x9 array

provided =

```
[ [:a, :b, :c],  
  [:d, :e, :f],  
  [:g, :h, :i],  
  [:j, :k, :l],  
  [:m, :n, :o],  
  [:p, :q, :r],  
  [:s, :t, :u],  
  [:v, :w, :x],  
  [:y, :z, :A] ]
```



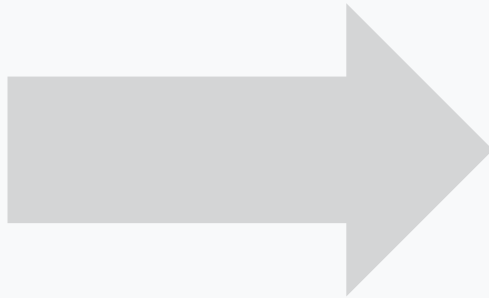
desired =

```
[ [:a, :e, :g],  
  [:l, :n, :q],  
  [:t, :w, :z] ]
```

Creating the Structure

```
provided =
```

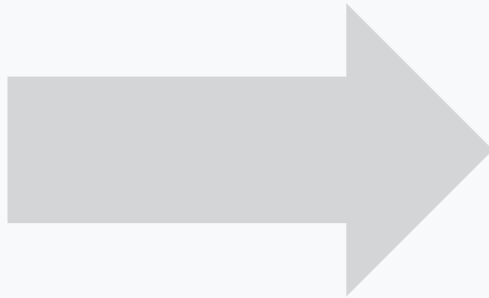
```
[ [:a, :b, :c],  
  [:d, :e, :f],  
  [:g, :h, :i],  
  [:j, :k, :l],  
  [:m, :n, :o],  
  [:p, :q, :r],  
  [:s, :t, :u],  
  [:v, :w, :x],  
  [:y, :z, :A] ]
```



Creating the Structure

provided =

```
[ [:a, :b, :c],  
  [:d, :e, :f],  
  [:g, :h, :i],  
  [:j, :k, :l],  
  [:m, :n, :o],  
  [:p, :q, :r],  
  [:s, :t, :u],  
  [:v, :w, :x],  
  [:y, :z, :A] ]
```



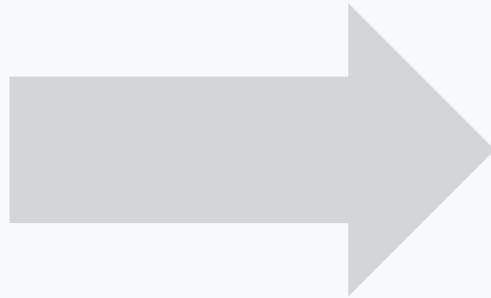
desired =

```
[ [nil, nil, nil],  
  [nil, nil, nil],  
  [nil, nil, nil] ]
```

Creating the Structure

provided =

```
[ [:a, :b, :c],  
  [:d, :e, :f],  
  [:g, :h, :i],  
  [:j, :k, :l],  
  [:m, :n, :o],  
  [:p, :q, :r],  
  [:s, :t, :u],  
  [:v, :w, :x],  
  [:y, :z, :A] ]
```



desired =

```
[ [:a, :e, :g],  
  [:l, :n, :q],  
  [:t, :w, :z] ]
```

Creating the Structure

- Start by creating the structure
- Then populate the result
- Finally manipulate the input

Creating the Structure

- Create and populate the desired structure

```
desired = Array.new(3) { Array.new(3) }
```

Creating the Structure

- Create and populate the desired structure

```
desired = Array.new(3) { Array.new(3) }
```

```
desired.map!.with_index do |row, row_index|  
  row.map!.with_index do |column, column_index|  
    provided.each_slice(3).to_a[row_index][column_index].sample  
  end  
end
```

Creating the Structure

- Refactor code for clarity

```
desired = provided.map(&:sample).each_slice(3).to_a
```


Ambiguity in Complex Arrays

- How do you access elements in an array?

Ambiguity in Complex Arrays

- How do you access elements in an array?

by index

Ambiguity in Complex Arrays

- Indexes are poorly named variables

Ambiguity in Complex Arrays

team			
number	name	position	points per game
12	Joe Schmo	Center	[14, 32, 7, 0, 23]
9	Ms. Buckets	Point Guard	[19, 0, 11, 22, 0]
31	Harvey Kay	Shooting Guard	[0, 30, 16, 0, 25]
18	Sally Talls	Power Forward	[18, 29, 26, 31, 19]
22	MK DiBoux	Small Forward	[11, 0, 23, 17, 0]

Ambiguity in Complex Arrays

```
team = [ ["number", "name", "position", "points per game"],  
         [12, "Joe Schmo", "Center", [14, 32, 7, 0, 23]],  
         [9, "Ms. Buckets", "Point Guard", [19, 0, 11, 22, 0]],  
         [31, "Harvey Kay", "Shooting Guard", [0, 30, 16, 0, 25]],  
         [18, "Sally Talls", "Power Forward", [18, 29, 26, 31, 19]],  
         [22, "MK DiBoux", "Small Forward", [11, 0, 23, 17, 0]] ]
```

Ambiguity in Complex Arrays

- How do I get the data for Sally Talls?

Ambiguity in Complex Arrays

```
team = [{"number", "name", "position", "points per game"},  
        [12, "Joe Schmo", "Center", [14, 32, 7, 0, 23]],  
        [9, "Ms. Buckets", "Point Guard", [19, 0, 11, 22, 0]],  
        [31, "Harvey Kay", "Shooting Guard", [0, 30, 16, 0, 25]],  
        [18, "Sally Talls", "Power Forward", [18, 29, 26, 31, 19]],  
        [22, "MK DiBoux", "Small Forward", [11, 0, 23, 17, 0]]]  
  
sally_talls = team[4]
```

Ambiguity in Complex Arrays

```
team = [ ["number", "name", "position", "points per game"],  
         [12, "Joe Schmo", "Center", [14, 32, 7, 0, 23]],  
         [9, "Ms. Buckets", "Point Guard", [19, 0, 11, 22, 0]],  
         [31, "Harvey Kay", "Shooting Guard", [0, 30, 16, 0, 25]],  
         [18, "Sally Talls", "Power Forward", [18, 29, 26, 31, 19]],  
         [22, "MK DiBoux", "Small Forward", [11, 0, 23, 17, 0]] ]
```

```
sally_talls = team.find { |player| player[1] == "Sally Talls" }
```


Ambiguity in Complex Arrays

```
team = [{"number", "name", "position", "points per game"},  
        [12, "Joe Schmo", "Center", [14, 32, 7, 0, 23]],  
        [9, "Ms. Buckets", "Point Guard", [19, 0, 11, 22, 0]],  
        [31, "Harvey Kay", "Shooting Guard", [0, 30, 16, 0, 25]],  
        [18, "Sally Talls", "Power Forward", [18, 29, 26, 31, 19]],  
        [22, "MK DiBoux", "Small Forward", [11, 0, 23, 17, 0]]]
```

Ambiguity in Complex Arrays

- What position does Ms. Buckets play?
- What number does Harvey Kay wear?
- How many points did Joe Schmo score in Game 3?

Ambiguity in Complex Arrays

- What values do these return?

```
team[2][0]  
team[5][3][0]  
team[3][2]
```

Hashes Provide Informative Labels

```
hash_team = {  
  "Joe Schmo" => { number: 12, position: "center",  
                  "points per game" => [14,32,7,0,23]},  
  
  "Ms. Buckets" => { number: 9, position: "Point Guard",  
                    "points per game" => [19,0,11,22,0]},  
  
  "Harvey Kay" => {number: 31, position: "Shooting Guard",  
                  "points per game" => [0,30,16,0,25]},  
  
  "Sally Talls" => {number: 18, position: "Power Forward",  
                  "points per game" => [18,29,26,31,19]},  
  
  "MK DiBoux" => {number: 22, position: "Small Forward",  
                  "points per game" => [11,0,23,17,0]}  
}
```

Hashes Provide Informative Labels

- Which is more comprehensible?

```
team[2][0]  
team["Ms. Buckets"]["number"]
```

Optimal Solution Maybe

An Array of Hashes

```
team = [  
  {"Joe Schmo" => { number: 12, position: "center",  
                    "points per game" => [14,32,7,0,23] }},  
  
  {"Ms. Buckets" => { number: 9, position: "Point Guard",  
                     "points per game" => [19,0,11,22,0] }},  
  
  {"Harvey Kay" => { number: 31, position: "Shooting Guard",  
                    "points per game" => [0,30,16,0,25] }},  
  
  {"Sally Talls" => { number: 18, position: "Power Forward",  
                     "points per game" => [18,29,26,31,19] }},  
  
  {"MK DiBoux" => { number: 22, position: "Small Forward",  
                   "points per game" => [11,0,23,17,0] }},  
]
```

Data Structs Wrapup

Questions