

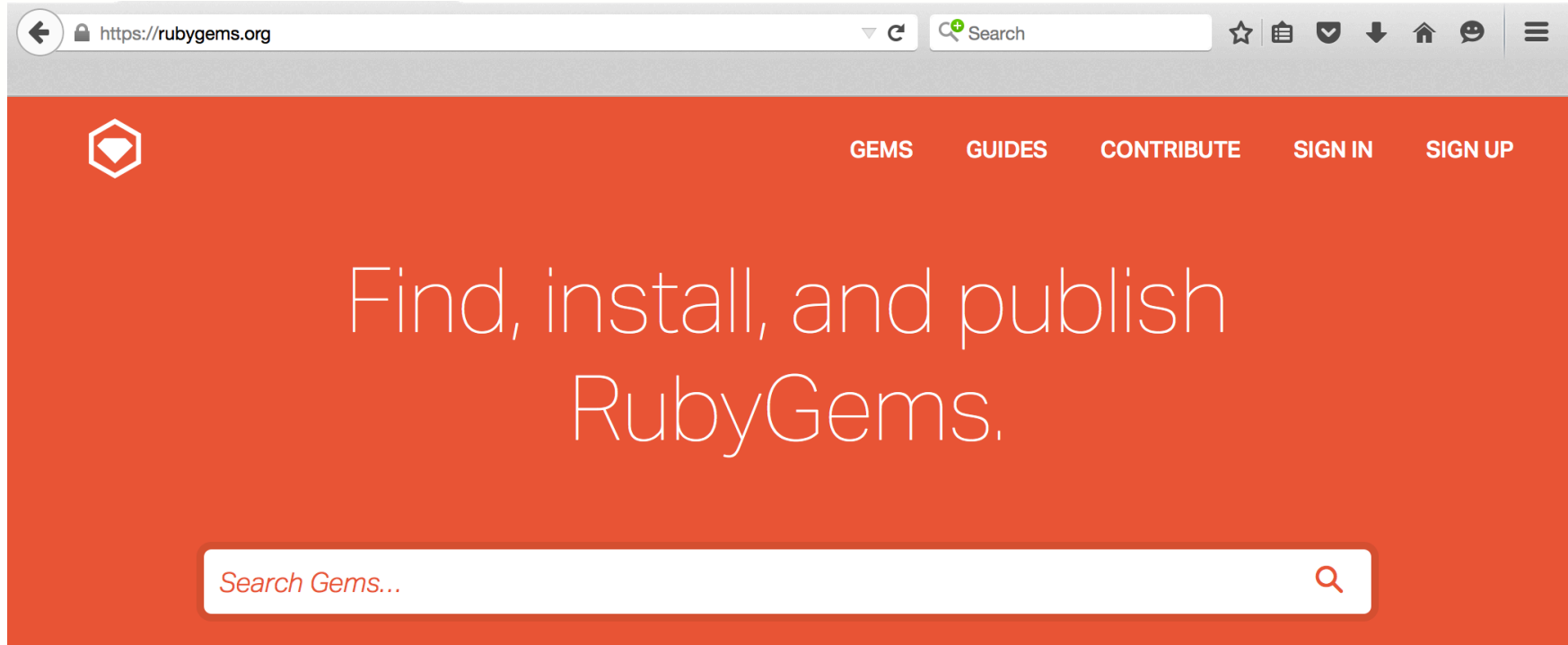
In this lecture, we will discuss...

- ✧ Ruby gems
- ✧ How to use the HTTParty Ruby gem



RubyGems

✧ Many gems (3rd party plugins / additions) for Ruby



RubyGems Package Manager

```
~$ gem
```

RubyGems is a sophisticated package manager for Ruby. This is a basic help message containing pointers to more information.

Usage:

```
gem -h/--help
```

```
gem -v/--version
```

```
gem command [arguments...] [options...]
```

Examples:

```
gem install rake
```

```
gem list --local
```

```
gem build package.gemspec
```

```
gem help install
```



Installing HTTParty

```
~$ gem list httparty
```

```
*** LOCAL GEMS ***
```

```
~$ gem install httparty
```

```
Fetching: httparty-0.13.5.gem (100%)
```

```
When you HTTParty, you must party hard!
```

```
Successfully installed httparty-0.13.5
```

```
Parsing documentation for httparty-0.13.5
```

```
Installing ri documentation for httparty-0.13.5
```

```
Done installing documentation for httparty after 0 seconds
```

```
1 gem installed
```



More Details on Gem Command

✧ Use `-d` option to **get more details**

```
~$ gem list httparty -d
```

```
*** LOCAL GEMS ***
```

```
httparty (0.13.5)
```

```
Authors: John Nunemaker, Sandro Turriate
```

```
Homepage: http://jnunemaker.github.com/httparty
```

```
License: MIT
```

```
Installed at: /Users/kalmanhazins/.rbenv/versions/2.1.2/lib/ruby/gems/2.1.0
```

```
Makes http fun! Also, makes consuming restful web services dead  
easy.
```



What Are Restful Web Services?

- ✧ **Simple** web services **implemented using HTTP** (and principles of REST) that:
 1. Have a **base URI**
 2. Support a **data exchange format** like XML or JSON (and possibly others)
 3. Support a **set of HTTP operations** (GET, POST etc.)






HTTParty Gem


- ✧ Restful web services **client**
- ✧ **Automatic parsing of JSON and XML into Ruby hashes**
- ✧ Provides **support** for:
 - Basic http authentication
 - Default request query parameters



Lots of Restful APIs Out There...

 www.programmableweb.com/apis 

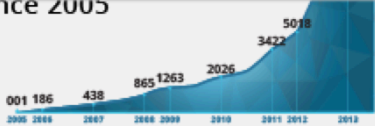
[Follow ProgrammableWeb to get API news and alerts as they break](#) [+Follow](#) [Share](#) [Sign In/Sign Up](#)

 **ProgrammableWeb** [API News](#) [API Directory](#) [For API Providers](#) [For Developers](#) [Listings](#) [Forum](#)

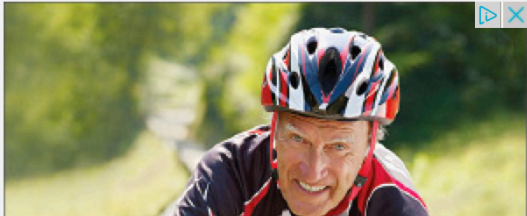
MOST POPULAR APIs

1. Facebook	Track this API	6. Pinterest	Track this API
2. Google Maps	Track this API	7. Bloomberg	Track this API
3. Fitbit	Track this API	8. Twitter	Track this API
4. LinkedIn	Track this API	9. FedEx	Track this API
5. TripAdvisor	Track this API	10. Instagram	Track this API

APIs Since 2005



Year	Number of APIs
2005	001
2006	186
2007	438
2008	865
2009	1263
2010	2026
2011	3422
2012	5018



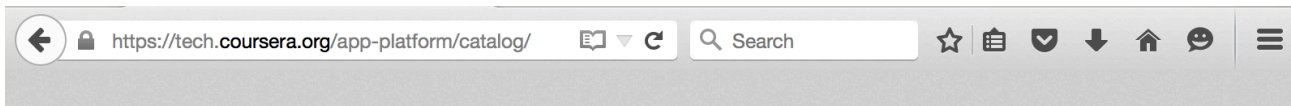


HTTParty Usage

- ✧ `include HTTParty` module
 - And you are good to go!
- ✧ Can specify:
 - `base_uri` for your **requests**
 - `default_params` (API developer key for example)
 - `format` to tell it **which format the data is in**



Coursera Restful API



Example requests

All of the catalog APIs are consistent and behave similarly. Sending a **GET** to the root of the resource will download the entire collection. To retrieve an individual element, use either a **GET** with an **id** query parameter, or append the id as a path parameter. For example, the following are equivalent:

```
1 curl https://api.coursera.org/api/catalog.v1/courses/2
2 curl https://api.coursera.org/api/catalog.v1/courses?id=2
```

All collections support multi-get with the following syntax:

```
1 curl https://api.coursera.org/api/catalog.v1/courses?ids=2,3
```

By default, only a minimal set of fields are included in response objects. To request more fields, include their names in a **fields** query parameter. For example, the



Coursera Restful API

Base URI

Parameters

```
{
  elements: [
    {
      id: 948,
      shortName: "webapplications",
      name: "Web Application Architectures",
      shortDescription: "Learn how to build and deploy modern web application architectures – applications that run over the Internet, in the \"cloud,\" using a browser as the user interface.",
      smallIcon: https://d15cw65ipctsr.cloudfront.net/8a/72e49851da4f79d1a3a3df7f840d5c/mooc\_logo.jpg,
      links: { }
    },
    {
      id: 117,
      shortName: "proglang",
      name: "Programming Languages",
      shortDescription: "Investigate the basic concepts behind programming languages, with an emphasis on the techniques and benefits of functional programming. Use the programming languages ML, Racket, and Ruby to learn how the pieces of a language fit together to create more than the sum of the parts. Gain new software skills and the concepts needed to learn new languages on your own.",
      smallIcon: https://d15cw65ipctsr.cloudfront.net/e1/644d37da2af639d0c9d1f4fca323c2/course\_logo.jpg,
      links: { }
    }
  ]
}
```

JSONView Browser Plugin (Available for Chrome and Firefox)



HTTParty Example

```
require 'httparty'
require 'pp'

class Coursera
  include HTTParty

  base_uri 'https://api.coursera.org/api/catalog.v1/courses'
  default_params fields: "smallIcon,shortDescription", q: "search"
  format :json

  def self.for term
    get("", query: { query: term})["elements"]
  end
end

pp Coursera.for "python"
```



HTTParty Example Output

```
[{"id"=>2760,  
  "shortName"=>"algorithmicthink1",  
  "name"=>"Algorithmic Thinking (Part 1)",  
  "shortDescription"=>  
    "Experienced Computer Scientists analyze and solve computational problems at a level of a  
    designed to train students in the mathematical concepts and process of \"Algorithmic Think  
  "smallIcon"=>  
    "https://d15cw65ipctsrr.cloudfront.net/70/6ff6409dd811e49bfb61275b434ba9/AlgTh_logo.jpg",  
  "links"=>{}},  
{"id"=>3048,  
  "shortName"=>"molecularevolution",  
  "name"=>"Deciphering Molecular Evolution (Bioinformatics IV)",  
  "shortDescription"=>  
    "In this course, we will see how evolutionary trees resolve quandaries from finding the o  
    from computational proteomics to test whether we can reconstruct Tyrannosaurus rex protei  
  "smallIcon"=>  
    "https://d15cw65ipctsrr.cloudfront.net/16/55b1b0f3c111e4a6a209002642cb9f/evo.jpg",  
  "links"=>{}},
```



Summary

- ✧ HTTParty makes it **extremely easy** to ingest Restful services, converting them to Ruby Hashes
- ✧ **JSON** and **XML** formats **supported**

What's next?

- ✧ Bundler

