## Week 2

- Ruby Classes, Objects, & Instances
- JSON APIs
- Rails 101
- MVC Architecture

## Get The Code

```
cd ~/dev/uc

git clone git://github.com/cspp52553/week2.git
```

x

## Demo + Lab

Goal: Display a list of Chicago landmarks

Each landmark should have two attributes:

- Name
- Admission Fee

Use an Array of Landmark instances.

Use puts statements to display the data.

x

## Consuming JSON APIs

**What is an API?**

**JSON notation**

**Converting JSON into a Ruby hash**

## What is an API?

**An Application Programming Interface**

**enables**

**computer-to-computer**

**communication.**

## What is an API?

**An <u>A</u>pplication <u>P</u>rogramming <u>I</u>nterface**

**is an agreement that enables**

**computer-to-computer**

**communication.**

## Broad Categories of APIs

**Platform**

**XML-RPC**

**SOAP**

**"RESTful"**

## Broad Categories of APIs

**Platform**

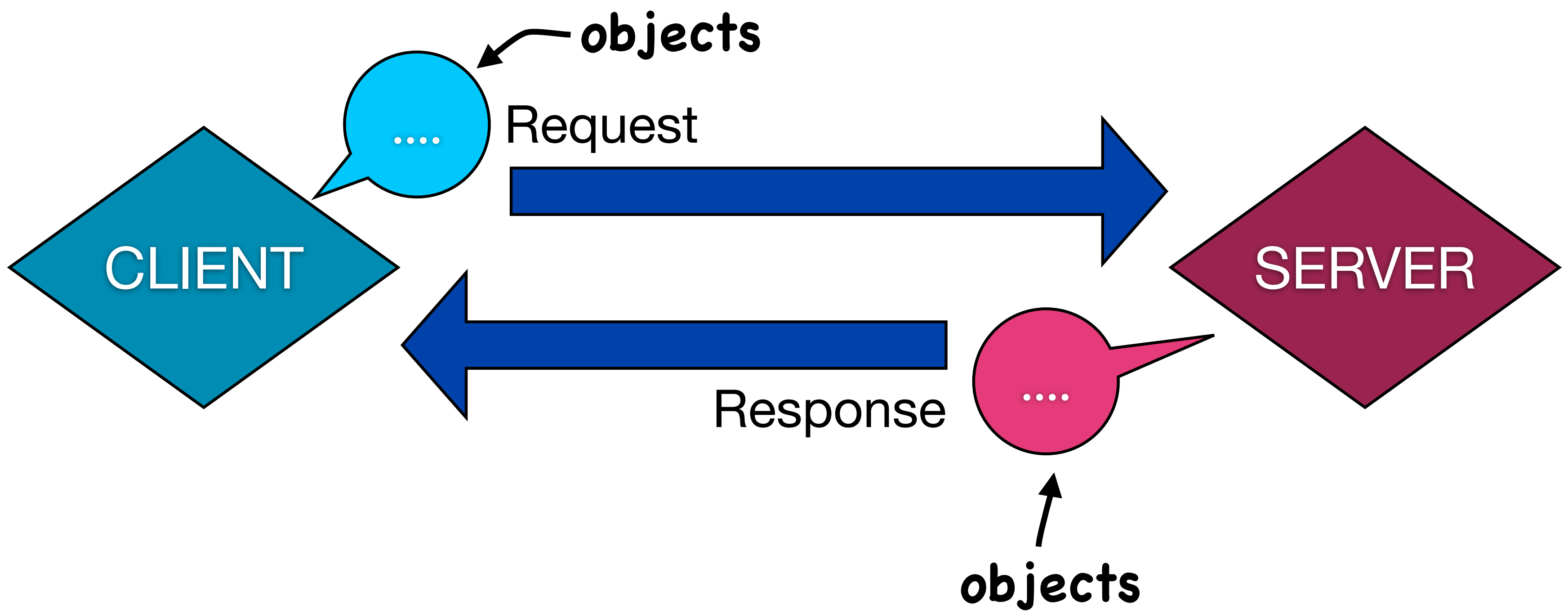
**-RPC**

**AP**

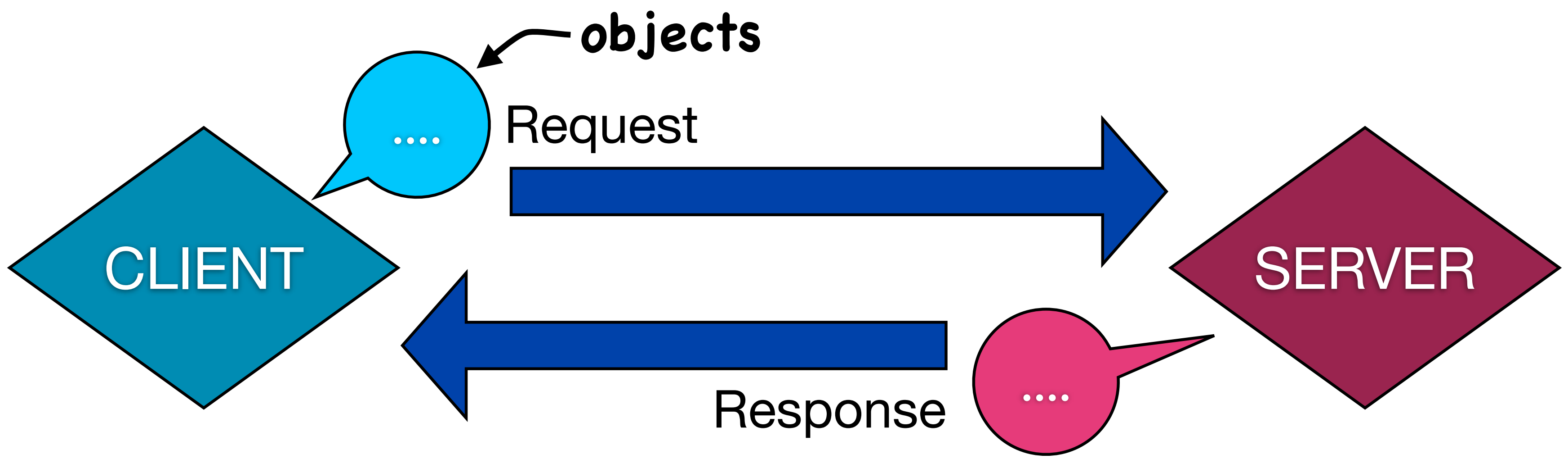
**"RESTful"**

Winner Winner
Chicken Dinner

## How Does An API Work?



# JSON APIs



**Javascript Object Notation (JSON)**

## Calling an HTTP API with Ruby

```
require 'json'
require 'open-uri'

data = open("http:...").read

h = JSON.parse(data)
```

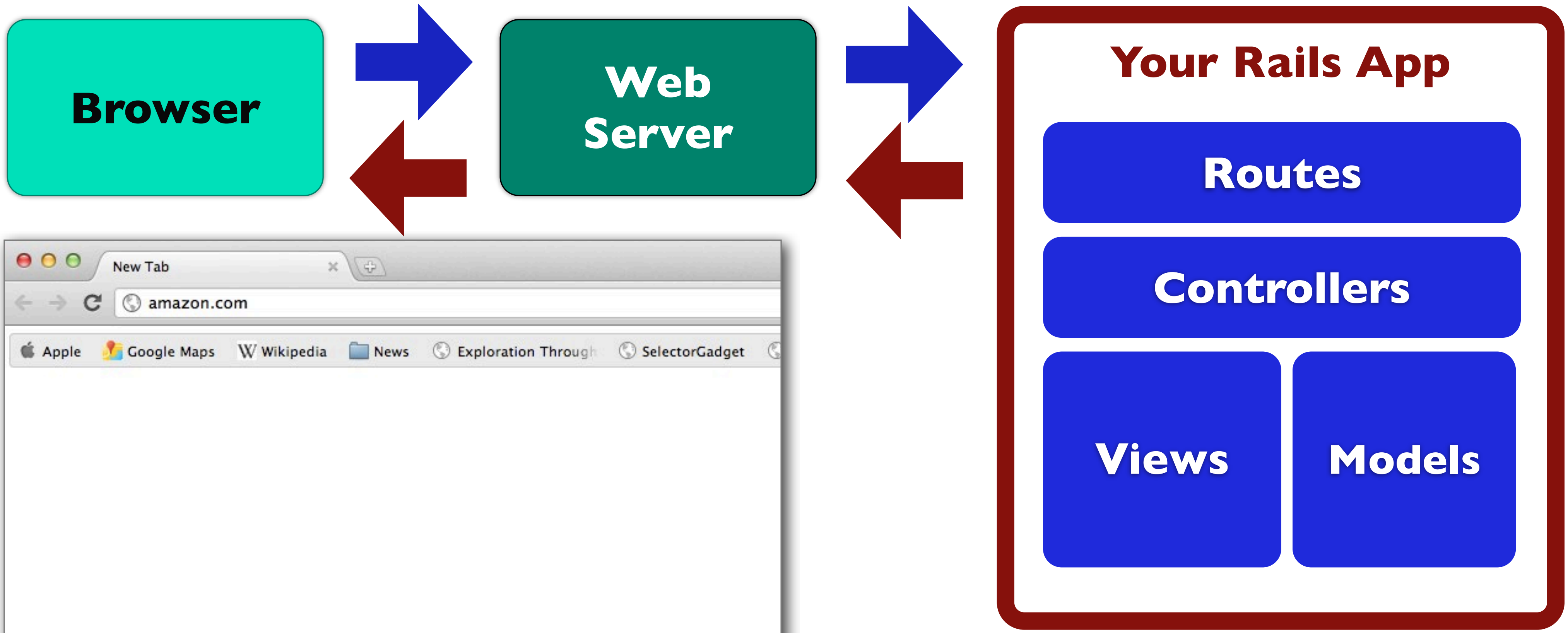
## Challenge: HTTP APIs

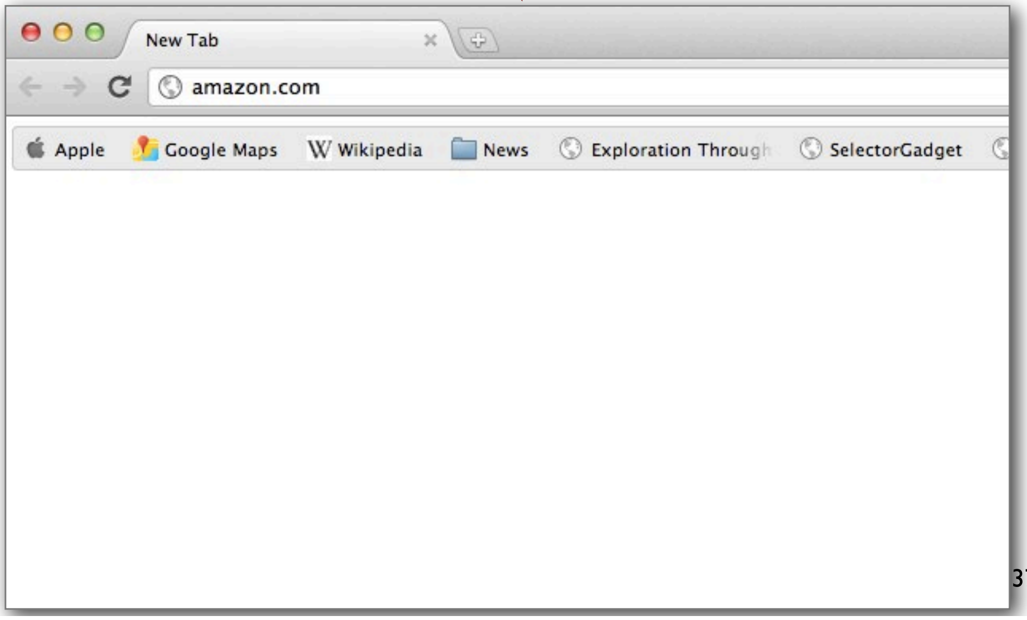
- require 'open-uri'
- require 'json'
- f = open("http://cspp52553.com/scrabble/hello.json")
- data = f.read
- Try to display the number of scrabble points

## HTTP

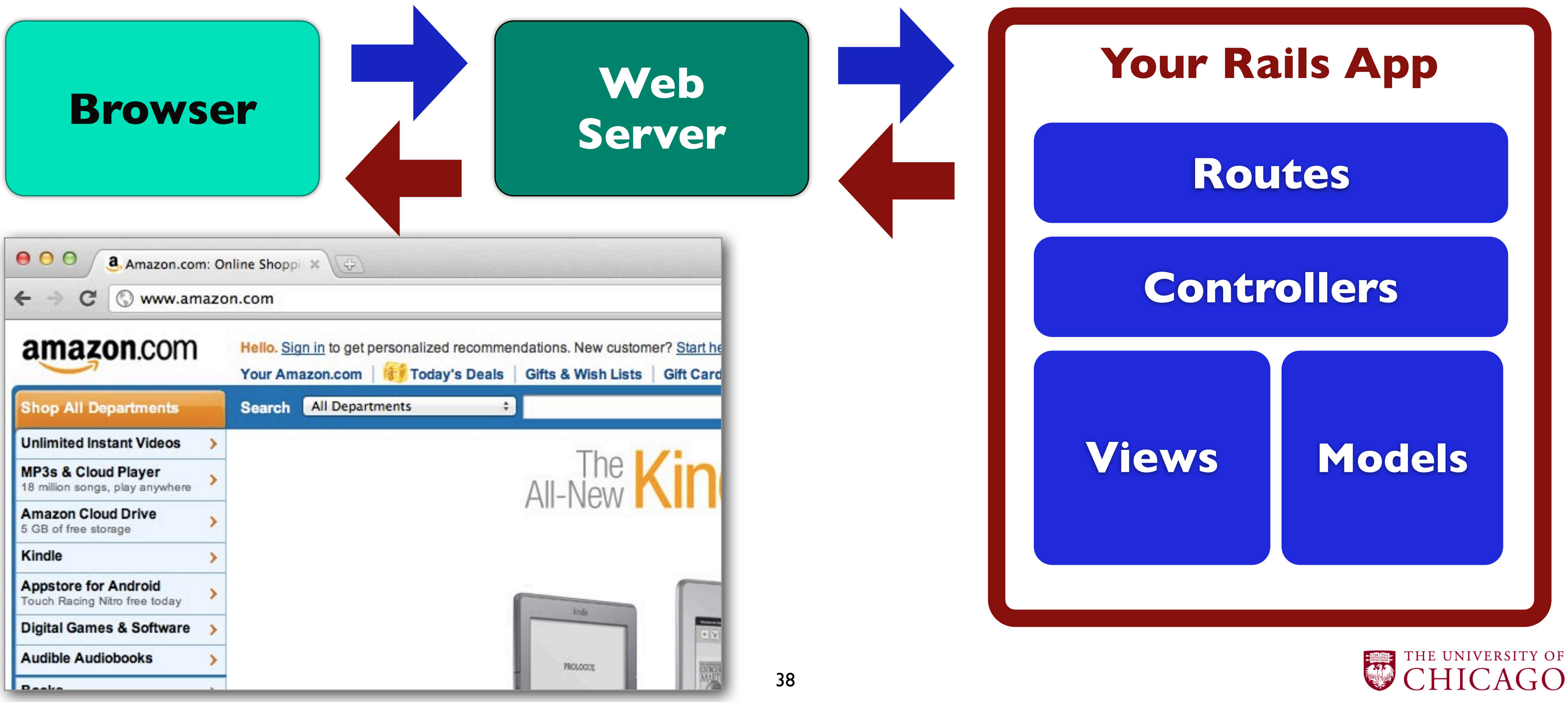
THE UNIVERSITY OF CHICAGO

# HTTP Overview

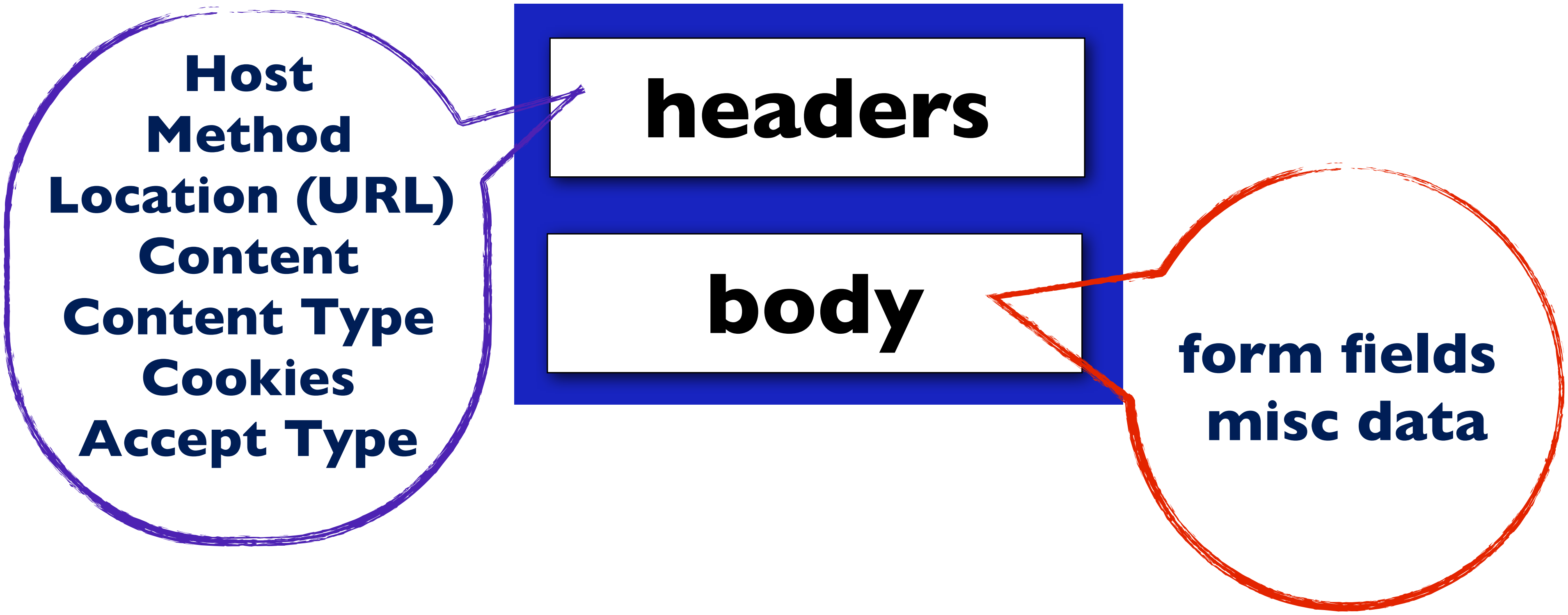




# HTTP Overview

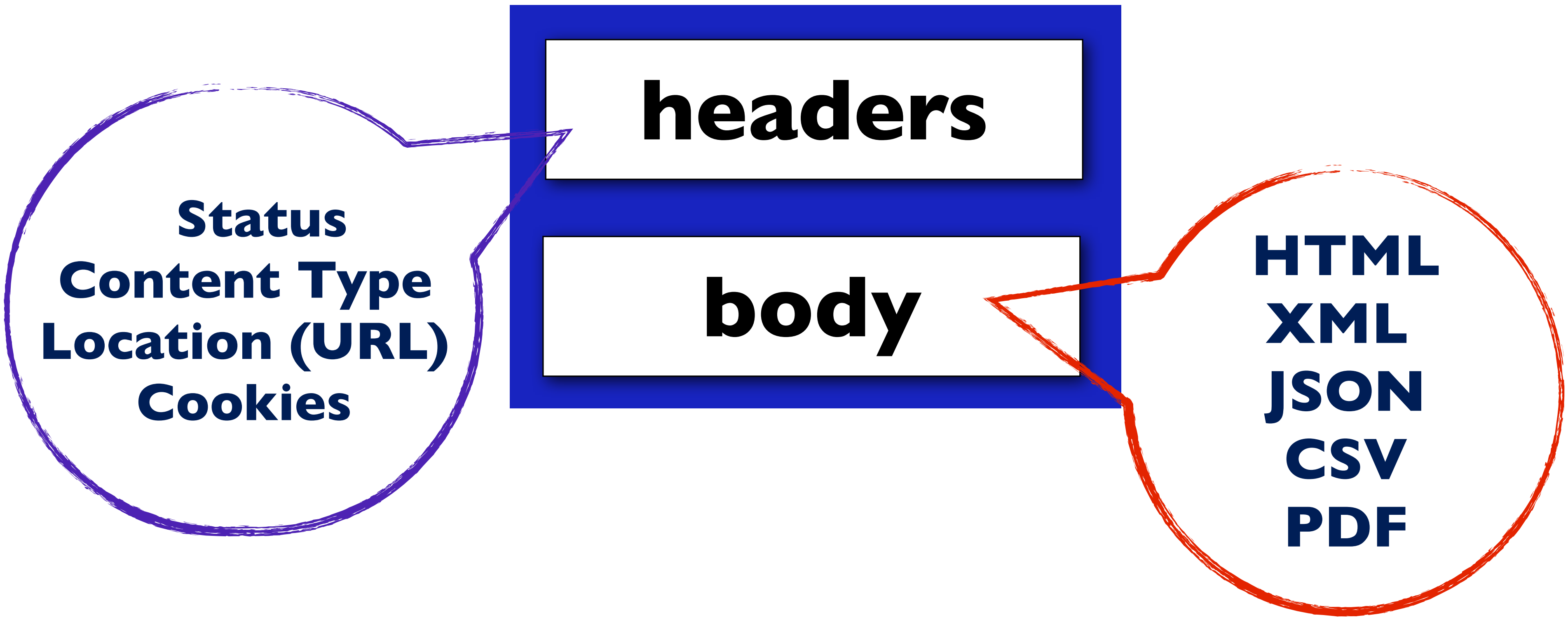


## HTTP Request



THE UNIVERSITY OF CHICAGO

## HTTP Response



## HTTP Playground

**Playing is the best way to learn!**

- curl
- Chrome Inspector
- *search for "http sniffer"*

## Ruby on Rails

RAILS

## Why Rails?

**Database-Backed Web Applications**

**Convention Over Configuration**

**Agile Development**

THE UNIVERSITY OF CHICAGO

## Why Rails?

STTCPW

YAGNI

DRY

SRP

## Why Rails?

**Responsibility**

## Rails QuickStart

1. cd ~/dev/uc/week2
2. rails new **myapp**
3. cd **myapp**
4. rails server
5. Go to http://localhost:3000

THE UNIVERSITY OF CHICAGO

## Domain Specific Languages

**omg lol i can haz ur speech**

## Rails: Routes

## Rails: Actions

## HTTP Response Headers

| HEADER | MEANING |
|---|---|
| Content-Type | Type of data |
| Location | Resource location |
| Status | Response code |
| Set-Cookie | Cookie data |

## HTTP Response Headers

**An action generates a response by using the render method or an HTML view template.**

## Rendering in HTTP Response

```ruby
def greeting
  render(:text => "Hello!", :status => 200)
end
```

**Generates an HTTP response with a body and header status code.**

## Rendering in HTTP Response

```
def greeting
  render(:text => "Hello!",
         :status => 301,
         :location => "http://www.apple.com)
end
```

**Generates an HTTP response with a body, header status code, and header location value.**

## Rails Views

## Rendering a view template

```
def greeting
  render 'greeting'
end
```

**Generates an HTTP response by using a *view template* named *greeting.html.erb***

## Rendering conventions

```ruby
def greeting
  render
end
```

**The filename can be omitted if it's the same as the name of the action method.**

## Rendering conventions

```ruby
def greeting
end
```

**The render statement can be omitted if you want to render a view with the same name as the action method.**

## Demo + Lab

Goal:  Create a web page of favorite things.

Use an HTML unordered list.

The URL should be:

http://localhost:3000/favorites

## Data-Driven Views

**Instance variables in action methods can be used inside the view template for that action!**

```
def greeting
  @salutation = "Wazzzzup!"
end
```

## Data-Driven Views

**We use embedded Ruby to insert logic and expression evaluation:**

```
<h1><%= @salutation %></h1>
```

## Demo + Lab

Goal:  Convert the view into a data template.

Create an array of items in the controller.

Then use ERb to *generate* an HTML unordered list.

## The RCAV Recipe

1. Define a route
2. Create a controller class
3. Create an action method
4. Create a view template

## Demo + Lab

Goal: Create a web page of your favorite photo.

Use the <img> tag.

The URL should be:

http://localhost:3000/photo

## Demo + Lab

**Goal: Use the *image_tag* view helper.**

## Demo + Lab

**Goal: Add links between each page.**

**Use <a> tags.**

## Demo + Lab

**Goal: Use the *link_to* view helper.**

## Rails Routes: Named Routes

**Naming a route makes it possible to do a reverse-lookup, and allows us to answer the question:**

***Given a route, what's the URL?***

## Rails Routes: Named Routes

**Use the `:as` option to set a "name" for a route.**

```
get '/my_favorites', :as => 'faves'
```

**Rails will synthesize two Ruby methods we can call whenever we need the URL.**

**Rails follows a convention for naming these two new methods.**

THE UNIVERSITY OF CHICAGO

## Rails Routes: Named Routes

```
get "/my_favorites",
    :controller => 'favorites',
    :action => 'index',
    :as => 'faves'

# We now have two methods named like this:
# faves_url
# faves_path
```