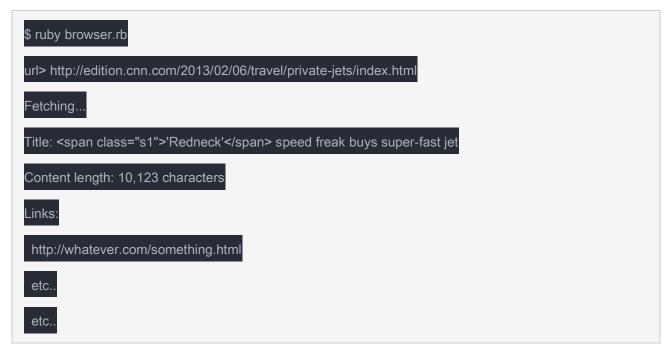
Time	~ 2 hours
Learning Goals	<ul> <li>Become more familiar with how the web works</li> <li>Learn about separation of concerns</li> <li>Learn about HTML parsing</li> </ul>

The git repo for this challenge: <a href="https://github.com/NextAcademy/dumbest-browser">https://github.com/NextAcademy/dumbest-browser</a> (remember to fork and clone!)

Now that you are more familiar with HTML, we're going to build the dumbest web browser in the world. It will work like this:



For this **challenge**, **we'll be using Ruby's built-**in <u>Net::HTTP</u> library to fetch the page and then use Nokogiri to parse the HTML the page returns.

We want you to start getting familiar with how the web works. Everything your program does, a browser has to do, too. We also want you to learn about <u>Separation of Concerns</u>. What responsibilities does your program have to fulfill? What classes do you need to fulfill those responsibilities? And where are the boundaries between the classes - how do they communicate with each other?

## Objectives

Listing the Responsibilities

Start by listing the core responsibilities of your app. Some of these responsibilities include:

- Displaying a prompt for the user
- Parsing user input
- Fetching a web page
- Extracting the relevant information from the web page
- Displaying the relevant information in a user-friendly way
- etc.

There are more - make sure you list them out! Next, group these responsibilities into "concerns". Prompting a user for input and taking the appropriate action might be one concern, for example. Fetching a URL and returning an easy-to-display data structure might be another.

## Toy Code

We will now learn how to fetch a web page. First, read about Ruby's <u>URI</u> module. By requiring 'uri' at the top of your Ruby program, you can use parse with a URL:

require 'uri'

uri = URI.parse('http://www.cnn.com/2013/02/06/travel/private-jets/index.html')

This gets the <u>URI</u> of the given URL. Then using the <u>Net::HTTP</u> module, get the HTML object from the URI. You can do this by:

require 'net/http'
html = Net::HTTP.get\_response(uri)

Here's a good <u>Net::HTTP</u> cheatsheet to help you. What kind of objects do <u>Net::HTTP</u> methods return? Should your program expose those objects directly?

*Hint*: No. Wrap them up in objects that more directly express what your code *does* rather than how its implemented.

Playing around with Nokogiri

First, make sure the **nokogiri** gem is **installed**. **We'll use this to** parse the HTML file. You can test this by running **irb** and typing

require 'nokogiri'

If you get an error that means Nokogiri is not installed. Install it by running this command:

## \$ gem install nokogiri

You'll want to have the <u>Nokogiri documentation about parsing a HTML document</u> available. In your page.rb file, your code would look something like this:

## body = Nokogiri::HTML(html.body)

What does the Nokogiri object itself look like? Don't worry about having to wade through its innards, but reading <u>Parsing HTML with Nokogiri</u> from <u>The Bastard's Book of Ruby</u> can give you a feel for how Nokogiri works.

Object-Oriented Code

Start with a Page class that has the following driver code:



See the original gist for where to put the "driver" code. You should have a Browser class that acts as the "brain" rather than lots of code living outside of a class.

Don't forget once everything is working, we want to have something that looks like this (as mentioned at the top of the challenge):

