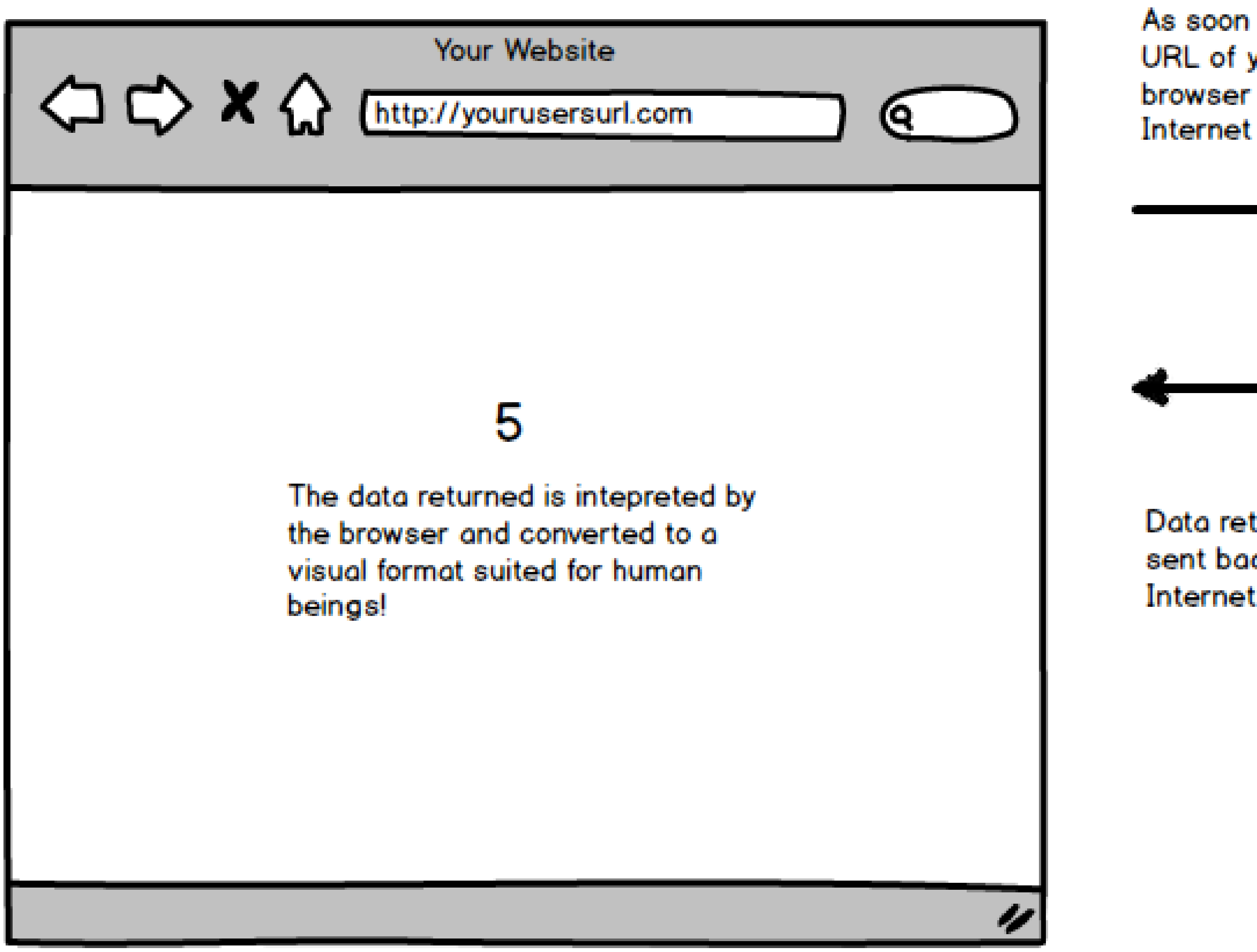
Time	~ 1 - 2 hours (including lecture time)
Learning Goals	• How do browsers (Chrome, Safari, IE, Firefox) and the web se work? • What is a Web Framework? • Understanding the MVC architecture • Understanding Gems and Bundler • Understanding Open Source Software

24:52

PART 1: Understanding the web and the HTTP protocol

See how data flows! A bigger version can be found here.


Your Website
http://yourusersurl.com
5
The data returned is intepreted by the browser and converted to a visual format suited for human beings!
As soon
URL of y
browser
Internet
Data ret
sent bac
Internet

All the arrows indicate data flow via what we term as Hypertext Transfer Protocol (HTTP). A protocol essentially specifies how a request should be formatted. There are many other protocols out there such as FTP, HTTPS, and SSH. HTTP makes up a huge component of the Internet.

Your job as a developer is to write code that responds to different URLs at step 3. Your code should respond with bytes of data to your user's browser to interpret and convert into something visual. The data that your app/code returns are most typically HTML files. HTML just like .doc or .xls is just a type of file with data that designated software can interpret and render (browser interprets HTML files, Excel interprets .xls files, Word interprets .doc files).

Key things to know:

1 The web request/response cycle
2 What is a client and what is a server
3 GET and POST requests

By the way, this is a good guide to help you get a good grasp of the web request/response cycle!

PART 2: What is a Web Framework?

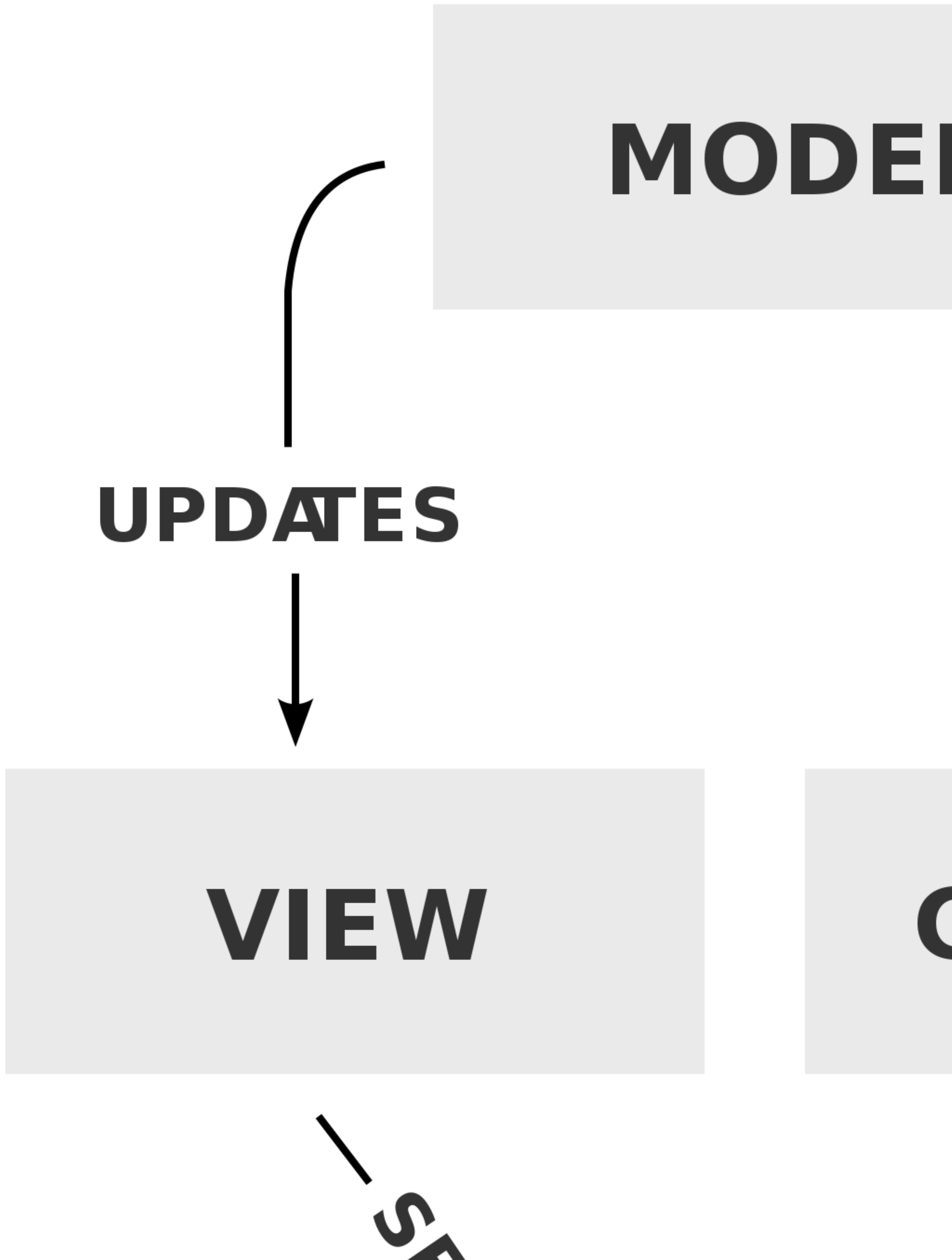
Who wants more work to achieve a similar result? That's the point of frameworks and libraries! In Ruby, we also call any of such reusable libraries Gems.

There's a lot of work to build any web app. And a lot of times, they are always the same kind of 'work'. As a result, a bunch of developers came together and build web frameworks (there are many web frameworks, you can build one too!). So, what are they? They are essentially a bunch of code written to handle the most common task we need to do like translating a HTTP request to only the information we need to work with, coding up forms etc.

We, and most developers, will use these code already written for us to write even more custom logic that fits our use case!

In Ruby world, Ruby-on-Rails is the most widely used framework. Other popular frameworks in Ruby include Sinatra, Cuba, Volt, Lotus. Other languages have their own libraries too. Every framework has its own strengths and purpose. Don't worry about this first. Know that you will be able to build most apps by the end of this program. Focus on learning how to be a developer instead of learning frameworks at this stage. When you are a good developer with strong fundamentals, new frameworks is just child's play (well, not entirely, but you get it :))
In this bootcamp, we will be using the Sinatra and Rails frameworks to build your apps. You will be building a Bitly and Quora clone with Sinatra, and AirBnb with Rails.
Key things to know:
1. What is a web framework?
2. Why do we use a framework?

PART 3: What is MVC?
The **M**odel, **V**iew and **C**ontroller (MVC) is a way to organize what code goes into which files/folders. In fact, we could have written our entire code base in one single file and have things work. But, imagine building any semi complex app with thousands of lines of code! It would be incredibly difficult to make any changes and look for the line of the code that causes a bug that you want to fix.
We split up code pertaining to each function to its respective files. And that's how you get an MVC architecture! Don't worry about what goes into what for this challenge, you'll get a hang of it over time. Just ask yourself everytime you write some code. Does this code affect how my web pages look (view) or will my code manipulate the data I retrieved from the database (model) or does it just serve the purpose of connecting the view with the database (controller)?
Here's a simple diagram to give you an idea of the relationship between the Models, Views and Controllers.


MODEL
UPDATES
VIEW

Key things to know:
1 What is MVC?
2 Why do we use an MVC architecture?

PART 4: Gems and Bundler

Gems are ways to package up code for other people or even just yourself to reuse easily. Each Gem usually has its own very specific purpose such as `communicating with twitter` or `connect to database` or `handle logging in of user` etc. It can be for any reason really. Think of the gems that you have used so far in the past few weeks. Did it make life easier for you? And gues what, next time, you can even write your own Gem!
Bundler is a piece of software used by Ruby developers to manage the necessary Gems for a particular project. Let's say you build a project that uses 10 gems, how are you going to tell your teammate you need this 10 gems effectively? Well, bundler has something called a `Gemfile` in each project. The file will list down all the `Gems` and its version. All your teammate needs to do is download your project and run `bundle install` to retrieve all the `Gems` usually stored somewhere in the Internet and most commonly `rubygems.org`. Of course, it's more than just for your teammates, it's also for you to manage your production environment easily!
Key things to know:
1 What is a gem and what is the purpose of using a gem?
2 What is Bundler and how does it make gem management easier?

PART 5: Open-source Software

What is software? Sometimes we don't know anymore. But everything is software. A Gem is a software, Rails is a software, your app is software, everything is software these days. Your car brakes are software.
But there is a difference between open-source and proprietary. If you write an app and not share or allow anyone to use the code

except for you, that’s proprietary software. If you write some software and want to share with the world to see the code, use the code and love, that’s opensource software. Ruby is an open-source language. In other words, you can see how the Ruby language was written. Yes, even languages are written. Sinatra, Rails are all open-sourced. The .net and iOS frameworks, for example, are proprietary. You can use it for its designated purpose by Microsoft and Apple respectively. You can’t see how it’s implemented. That’s why many developers love open-source. They can improve it if they want to. They don’t need to call Microsoft and say hey, there’s something wrong with your software, please fix it.

Key things to know:

1 What is the difference between open-source and proprietary softwares?

Your Turn

Write down what you learn. It’s important to recap in words what you learn. It helps you understand and remember!