**Week 4 Day 3 Notes**

Every time you enter a URL in your browser, you're making a request for certain information to a certain server. When your page loads, it's because you've received the response from the server with the information you requested. The entirety of the web is designed on the concept of clients that make requests to servers and servers returning a response for the requested information. During this lecture, we'll cover how the web works and the different parts of the HTTP (Hypertext Transfer Protocol). We'll also cover how to get started writing our very first Ruby web application using Sinatra!

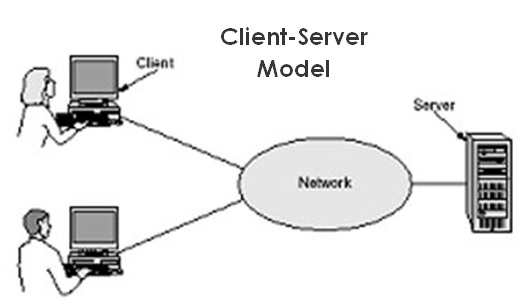
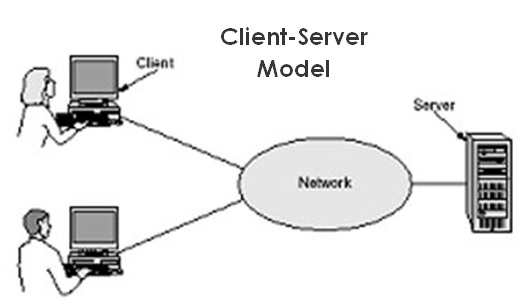
We’ve been working on fundamentals of programming, the basic ABCs, now we’re moving up a level. We’re learning the how to actually make something, make a web app. First hour of the lecture we’re focusing on the concepts of how the web works. You must understand these fundamentals.

* HTTP = Hypertext transfer protocol. Protocol = system for how things are done.
* The web is not the interent, the internet is much more than the web. Skype is part of the internet, but its not part of the web.
* HTTP is based on requests and responses.

**Request-Response Cycle**

* How information gets passed around on the web
* Client-server concept   
  --------Query------->

<-----Response----- (respones is result of attempt to fullfill your request)



### Parts of a request

* URL (very first thing you need to provide to send a request) (typing in an address in a web browser is an example of sending a request)
* Request method (GET request [asking for info], POST request [giving info])
* Headers (Everything that is not included in the body, like the writing on a package box. We’re not talking about a web header specfically)
* Body (sometimes)
* No status code

### Parts of a response

* Response status code (says yes we’ve got what you want, or here is an error message)
* Headers
* Body (most of the time)

## **Request Methods**

Since a URL represent the location of a particular set of information, the request methods let us indicate what we want to do with it. URL + HTTP verb.

Common request methods: (type of request you want to make)

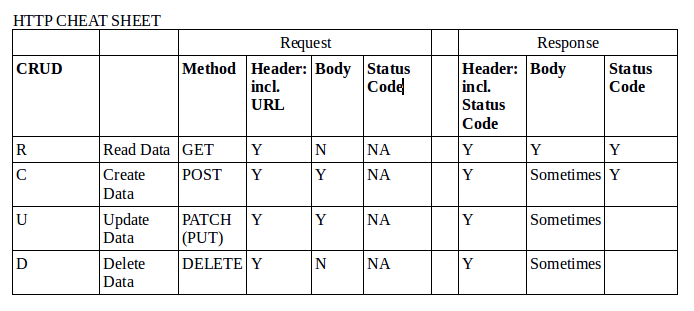
* GET (read data, used to fetch specific piece of data)
* POST (insert data, an example is sending out a tweet, creates data on the internet)
* PUT or PATCH (update data, or insert if a new id) (patch is new name that replaced put, example would be editing a forum post, requires a body)
* DELETE (delete data, such as deleting a tweet, does not require body)

There are more request methods but those are the most common ones, the rest are more obscure.

Web server’s job is to sit back and wait for requests. It may respond in HTML/CSS, or with a different language. Depends on the app.

Stateless means every request is a seperate request, web server does not remember where you have been. Each new requestion needs to provide the right context. Until “cookies” come into the picture.

**CRUD** --- Create, Read, Update, Delete



## **Response Codes**

Response codes let the browser know the status of the request, what happened.

Response code categories:

* 1xx: Informational (not really used)
* 2xx: Success
* 3xx: Redirection

301 Moved permanently  
302 Found

* 4xx: User Error

403 Forbidden (request but valid but server refusing)

* 5xx: Server Error   
  500 interal server error

http.cat (for funny cat images of response codes)

## **Sinatra** [www.sinatrarb.com](http://www.sinatrarb.com/)

## -Barebones web-server written for ruby, implements server-side of HTTP so you don’t have to. -Sinatra is a mini web-framework for Ruby we'll be using to learn the different parts of web development before we dive into Rails.

-Sinatra is a DSL (domain specific language) for quickly creating web apps in Ruby with minimal effort.

### -Over the course of your career you WILL be learning many langauges. So as a beginniner it does not make sense to think too much about what language you want to focus on and start with.

require 'sinatra' (must put in at the top of an .rb file)

-localhost:4567 (access local computer, we typed it in browser after putting in

/17-http-requests-and-responses$ ruby server.rb in the terminal.

-Upon loading the Hello World! Page created from our .rb file, the server in terminal recorded that our page was loaded:

127.0.0.1 - - [12/Jul/2017:11:16:12 -0400] "GET /favicon.ico HTTP/1.1" 404 469 0.0011

-favicon is the little icon on a webpage tab in a browser, in our case we didn’t define one so it gave us a 404. Browser tried to get a favicon but couldn’t. Server recorded that this happened.

-Sinatra is the little brother of rails, its way simpler.

The code we/Fred wrote at /bitmaker/lessons/wdi-june-2017/server.rb IS your server. Its sinatra code, but since its a dialect of ruby its a rb file.

* We can write HTML that will load in the sinatra rb file if we want, but it has to be in ‘ ‘ quotes. But then we can also put the HTML in a seperate file that we can link to (*see example syntax below*), that ideally should be in a subdirectory folder in our project folder.

Get ‘/hello’ do   
 send\_file ‘views/hello.html’  
end

**ROUTE**: Is simply the combination of a request method + a URL.  
Eg. Get ‘/’ do , get ‘/hello’ do

* You can have more routes than pages, if some of your routes direct to the same page.

<%= Time.now %> (result will show up on page)

<% \_\_\_\_ %> (result will not show up on the page)

^Ruby code contained in “alligator tags”, how to put ruby code in a html/erb file.

The two alligator opening/closing tags above are different, one has an equals sign, one doesn’t. There is a difference between them. The one with = will display result of code on the page.  
  
Ideally we want to put as little ruby as possible inside our HTML, because in a work envirornment you might have to hand the HTML code to a designer who won’t understand the Ruby code.

What is an ERB file? ERB stands for “embedded ruby”. Script written in ERB, a templating language for Ruby; may include any type of plain text or source code, but also includes Ruby ERB code that generates additional text into the resulting file when run with the ERB template engine. A .html.erb or .erb.html file is HTML with Ruby code embedded in; Rails will evaluate the Ruby to add content to the file dynamically, and will output a "pure" HTML file for rendering.

Read the sinatrarb.com getting started documentation.