PURPOSE:

A crash course of 3 hours/day one day a week for 6 weeks to introduce a person to the ideas behind web programming. This is not to teach a given language. This is not to teach a specific technology stack. This is an overview course to give a person the ability to understand HOW/WHY things work on the web.

There will be ~18 hours of instruction in total, and some of that needs to be for lab/homework/question time

Potential topics to cover:

- Brief history - (TCP - arpanet?, HTTP - cern) - purpose and motivation behind the need of network and heypermedia document sharing
- Network transfer of data (TCP/IP, packets, ports) - if this is about web programming I would include some crucial details in brief history since this is a really big area and I think it should be rather abstracted from the user
- HTTP transfer of data (stream of characters, separators, headers, body, footers) - put http verbs and status codes around here
- Individual webservers (Apache, IIS, nginx, lighttpd) - first motivation, why we need them, what are their functions and then proceed to specific vendors
- HTML
- CSS, styles and web design
- JavaScript?
- Flash and other plugin oriented technologies
- How CGI works
- How RoR/C# (MVC?) style apps work
- How AJAX works
- Why AJAX
- HTTP VERBS
- HTTP vs CRUD
- databases and their role
- types of databases (relational, nosql, newsql)
- the language of the web
- RFCs
- HTTP status codes and why we have them
- MVC principle/design pattern
- Single page applications

I would probably break these topics into some categories like core, http, databases, html/css/webdesign/flash, JS, frameworks, …

So you want to make a web application?

- Understand HTTP / TCP.
- Grab a HTTP server
- Define your first HTTP resource. Define what it will do. Think of the correct verb. Think of a sensible resource identifier. Read some RFCs
- Great we have a HTTP route, now what? We want some kind of data right? Pick a database, any database. Can't be bothered with a database? Use git or file system or in memory store
- Define some more HTTP routes, flesh out your CRUD, learn about REST.
- We have a nice little api we can query with restclient or curl, but it does shit all in the browser, what do browsers use? How do we render things?
- HTML! dive right into HTML5 and semantic elements for displaying data. Give them some kind of templating system for generating dynamic html files based on data from the HTTP api they wrote
- Browsers now render it awesome!
- Understands the ways people can talk to your HTTP server. Browser, AJAX, other applications, any HTTP client
- Understand mime types and sending different resources, xml, json, html, foobar, images, yada yada
- High level server-side architecture. MVC, 3 tier setup, yada yada