Project 7

Ajax Requests

Basic Ajax

Need a way for browser to communicate with server, possibly with user-entered data

Asynchronous Javascript and XML (not quite!!)

Client initiates the request

Needs to be fast

For this project

Write your own Ajax wrapper over XHR

More of a design based assignment, implementation is straight-forward

REUSABILITY IS A BIG PART

XHR is the basic entity you want to interact with to get your data

For this project

Dont worry too much about IE quirks (ActiveXObject etc.)

Client Side

Start with XMLHttpRequest object

The usual places on the web

Creating and using the object :

```
var myXHR = new XMLHttpRequest()
```

(TIP - Might want to persist this object instead of local var)

```
myXHR.open("Method_Type","URL",true/false)

myXHR.send() // actually send the request
```

Client side

Important parameters :

Type of request method - GET / POST etc.

URL - This is where the server will respond at

(URL can contain parameters)

(eg : "/state/filter?substring=ca")

Async flag - true (non-blocking) vs false (blocking)

Client Side

Q. How to know when response is available?

A. use onreadystatechange attribute!
myXHR.onreadychange = function() {.....}

Typically XHR will go through 5 states :
0 = uninitialized / just created
1 = typically after open
4 = response available

Client Side

Accessing current state value :

    if (myXHR.readystate == 4) {....}

What if I make a wrong URL call?

    check response status via myXHR.status *after* response is available

    status is usual 200, 403 etc..

Client Side

Getting actual content from response :

    myXHR.responseText

The text (HTML?) is generated by server

(More on server in a bit)

Set this text as HTML of your element of interest

Server Side

Treats incoming request like a normal request (controller, action etc.. using routes, accessing params)

Same MVC decomposition :-)

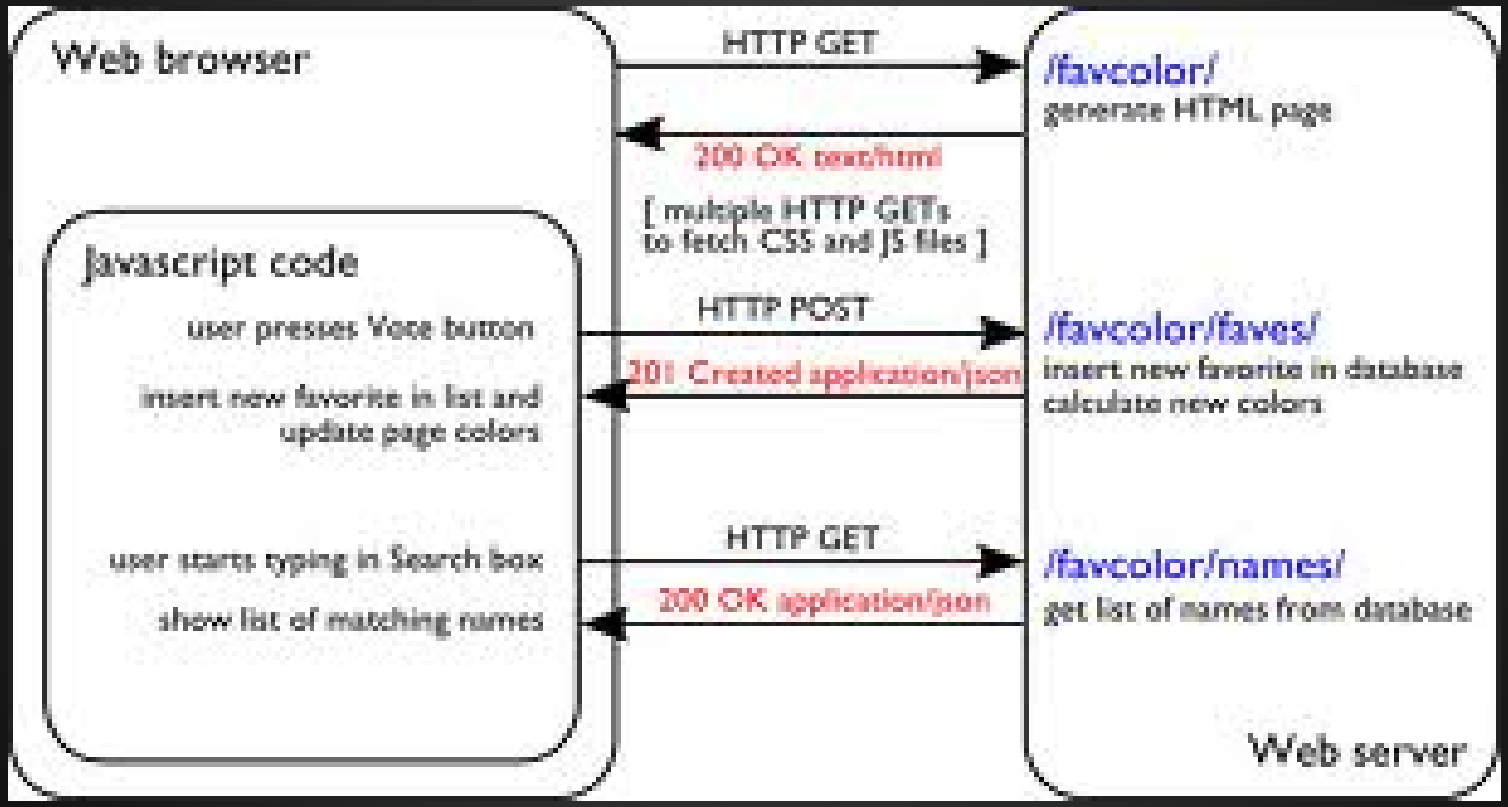
Sending data back :

- Generate your HTML - usual MVC
- Think partials!!

Sample server side

```
def filter
@substr = params[:substring]
@matchingstates = State.filter(@substr)
render :partial =>"filter"
end
```

Putting it all together
(IGNORE JSON)



Tips

You already know JavaScript. Just like
onmouseup, you'll find onKeyPress / onKeyUp useful

innerHTML is your friend
Status codes are good
layout can be supressed using render :layout => false

For reusablity, think what different requirements are, and what exceptions they might throw

For the enthusiasts

jQuery implementation of Ajax (very much involved, try skimming through it to get an idea)

Fun playing around with JSON data (remember responseText??)