AJAX & JSON

Code 301

WEEK 2

You will construct a model layer in your application that accesses and transforms persisted client-side data, conforming to common WRRC, FP, and CRUD conventions.

THE HTTP PARTY 🎉

HTTP

➤ Hypertext Transfer Protocol

➤ "Hypertext is structured text that uses logical links (hyperlinks) between nodes containing text."

➤ HTTP is the foundation of data communication for the World Wide Web.

➤ HTTP is the abstraction layer between your browser and lower-level the Internet communication protocols (TCP/IP).

POWER TO THE BROWSER!

➤ Browser technology powers the World Wide Web

➤ Let's take a peek under the hood!

➤ What powers browser?

MAGICAL CONVERSION: INPUT —> OUTPUT

???

???

URL (UNIFORM RESOURCE LOCATOR)

➤ Some URLs are nice:

http://twitter.com/codefellowsSEA

https://www.codefellows.org/blog.html?page=2

➤ Some URLs are not:

https://plus.google.com/u/0/110552115013326646668/posts

http://www.amazon.com/dp/B00VKIY9RG/
ref=s9_acsd_bw_dcd_odsbncat_c6_pd?
pf_rd_m=ATVPDKIKX0DER&pf_rd_s=merchandised-search-2

URL (UNIFORM RESOURCE LOCATOR): A BREAKDOWN

➤ Ugly or nice, they can all have the same parts:

➤ protocol: https://

➤ [sub]domain: www.codefellows.org
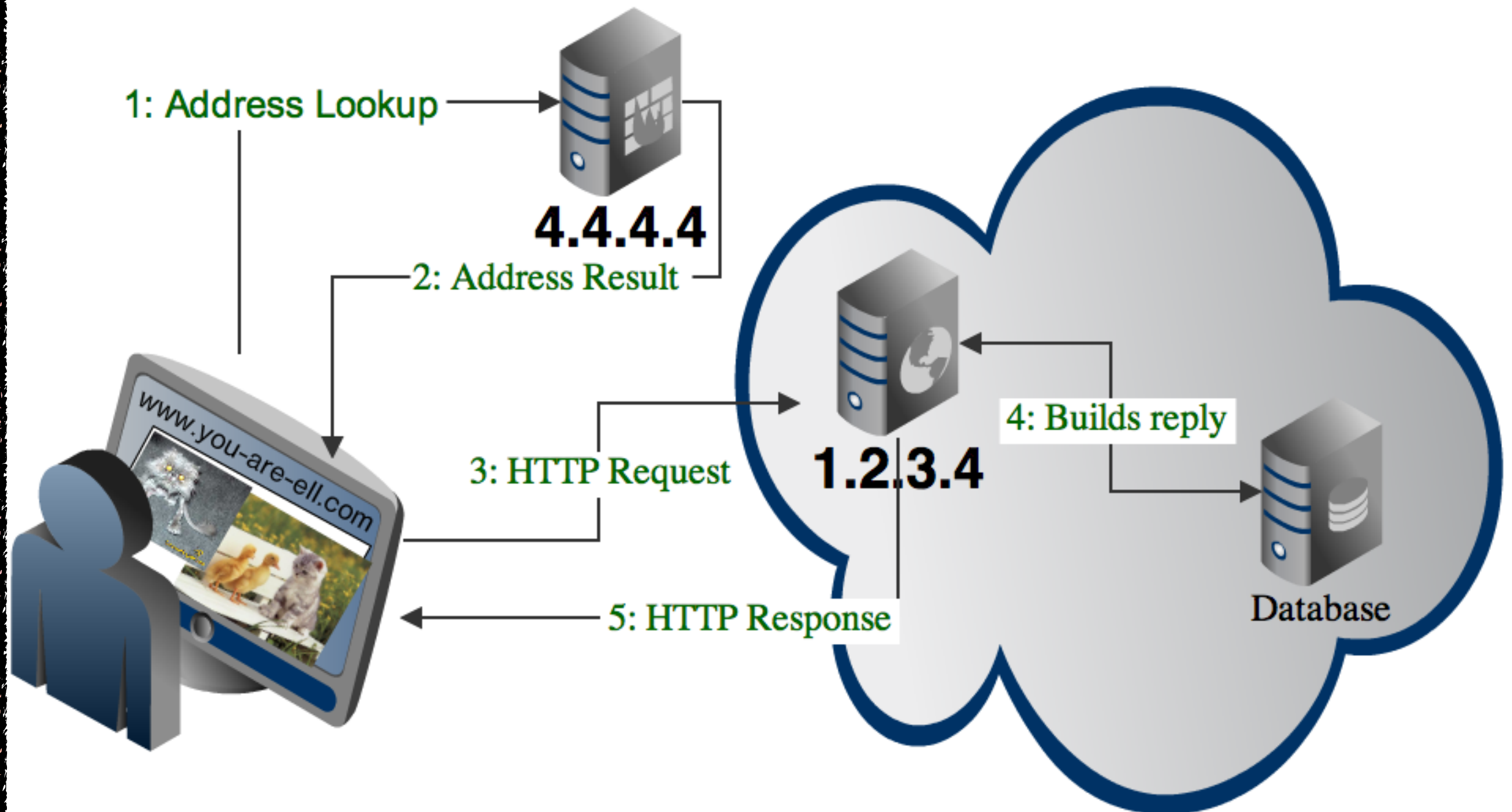
➤ path: /blog

➤ format: .html

➤ named anchor: #today

➤ parameters: ?page=2&admin=true

https://www.codefellows.org/blog.html#today?page=2&admin=true

Welcome to the HTTP Party!

THE DNS DANCE

➤ Talk to a known DNS Server

   ➤ Global index of domain names

➤ Give a name

➤ Get an IP address

➤ Now ready to ask for that web page!

➤ google.com = 216.58.193.78

➤ ping command!

THE HTTP PARTY

➤ The browser creates a HTTP Request Object

➤ HTTP Request has 3 parts:

  ➤ URL (twitter.com)

  ➤ Method (GET)

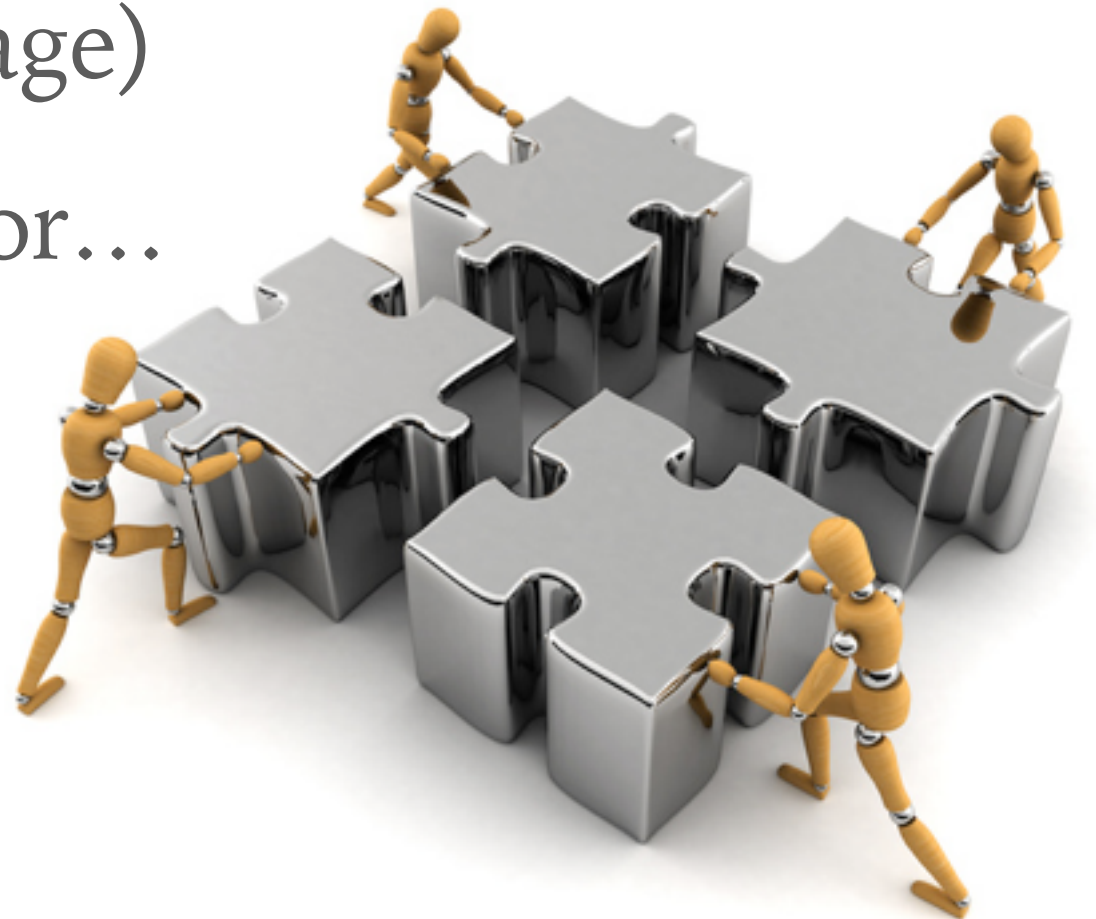 ➤ Headers (sender info: user agent, cookies, etc)

THE HTTP PARTY

➤ The server:

  ➤ receives the request

  ➤ builds a response

  ➤ sends it back to the client

THE HTTP PARTY: STATUS CODES

➤ An HTTP Response also has 3 parts:

  ➤ Status code (200, 301, 404, 500, etc)

  ➤ Headers (info about the server & file sent)

  ➤ Body (the content of the page)

    ➤ HTML, CSS, JavaScript, or…
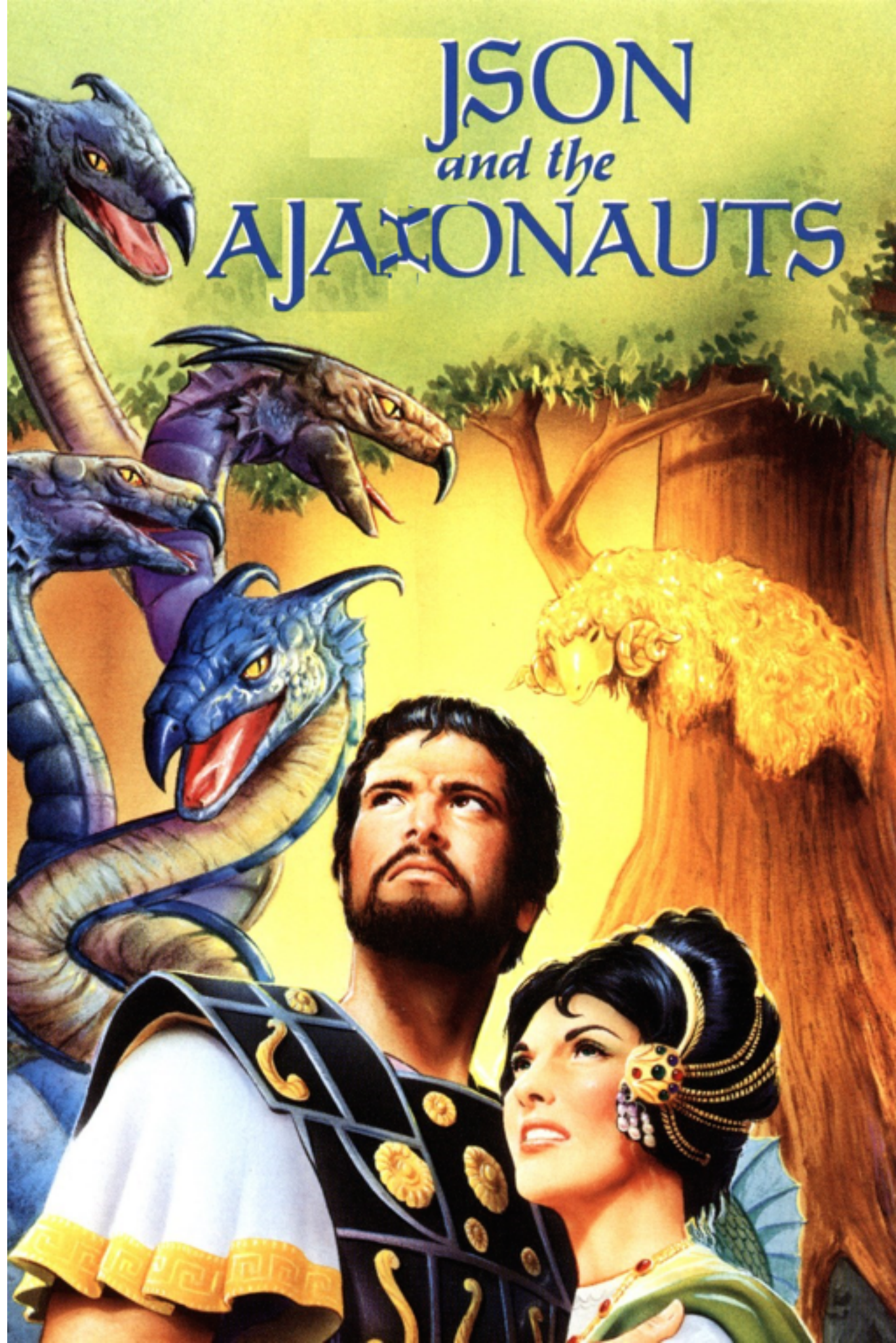
THE HTTP PARTY: STATUS CODES

➤ See: http://httpstatusdogs.com/

➤ See also: https://http.cat/

DEMO: TRY IT OUT

➤ Let's start with our Blog!

➤ Go to a site you are signed in to

➤ Inspect it: Explore the Inspector tabs

➤ Identify what is going back and forth

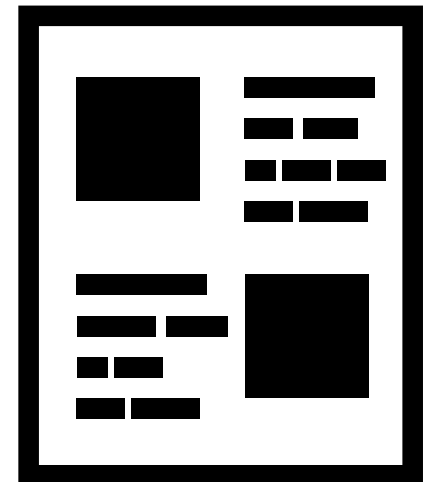➤ Figure out how to delete cookies. What happens?

JSON

JSON

➤ Looks like this:

```
1 {
2     "author": "Virginia Sawayn",
3     "title": "Navigating Solid State Multi-byte Monitors"
4 }
```

JSON: WHAT

➤ JavaScript Object Notation

➤ JS objects are lists of key-value pairs

JSON: WHY

➤ JSON is a standard way to **serialize** your objects

  ➤ "Dehydration for data structures"

  ➤ Reconstitute it later when needed

➤ Ate XML for breakfast

  ➤ More human readable

  ➤ More "object-oriented"

  ➤ Lighter

JSON VS. XML NOTATION

➤ In most situations, JSON is more compact than XML

➤ JSON

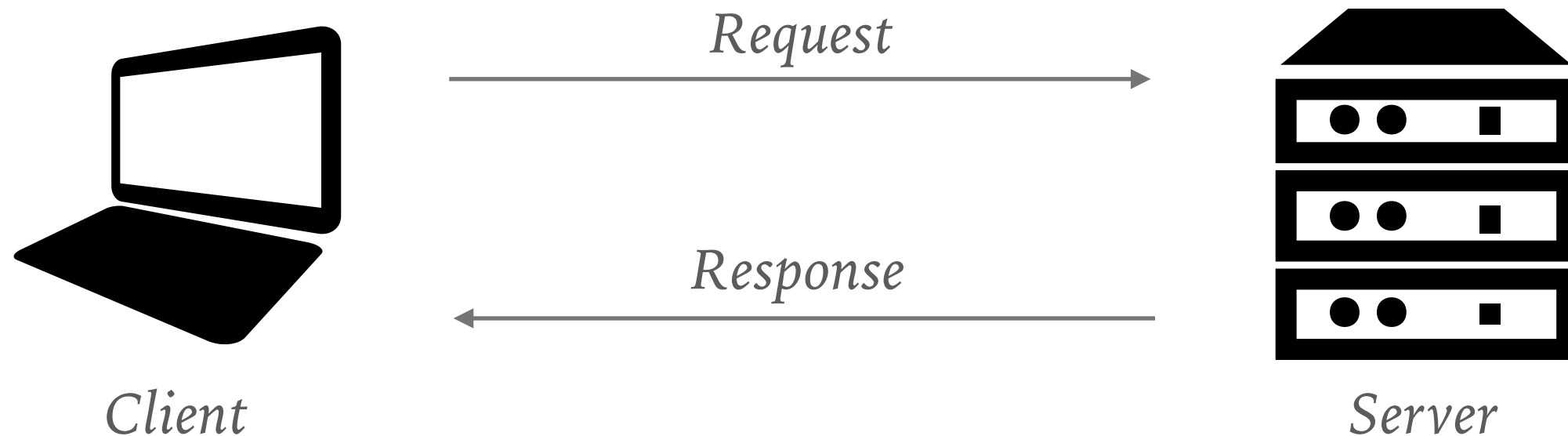```
1 {
2   "company": "Volkswagen",
3   "name": "Vento",
4   "price": 800000
5 }
```
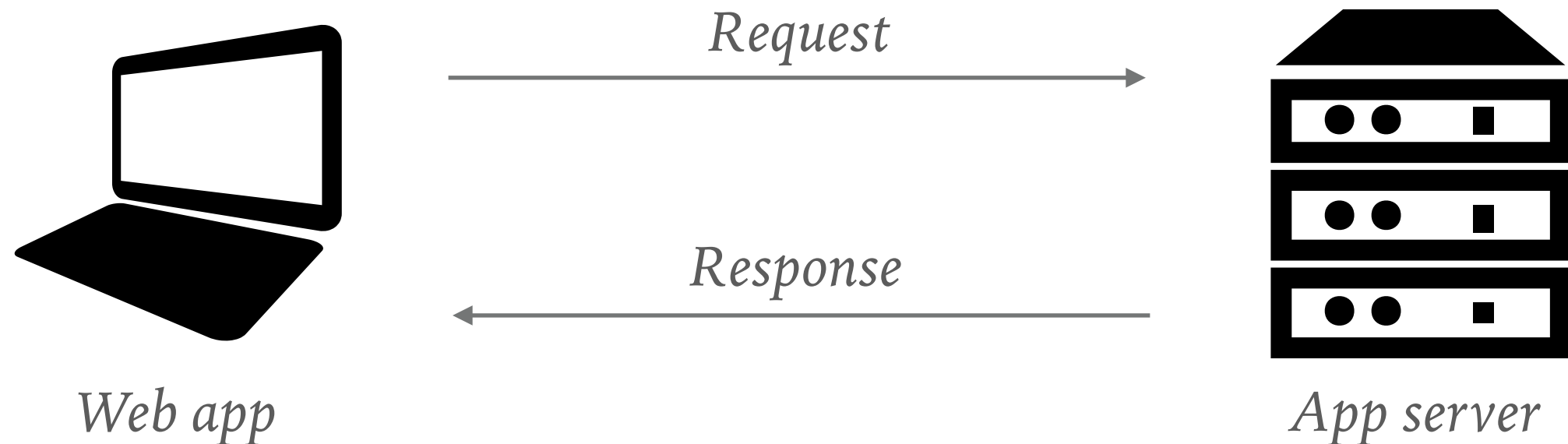
➤ XML

```
1 <car>
2   <company>Volkswagen</company>
3   <name>Vento</name>
4   <price>800000</price>
5 </car>
```
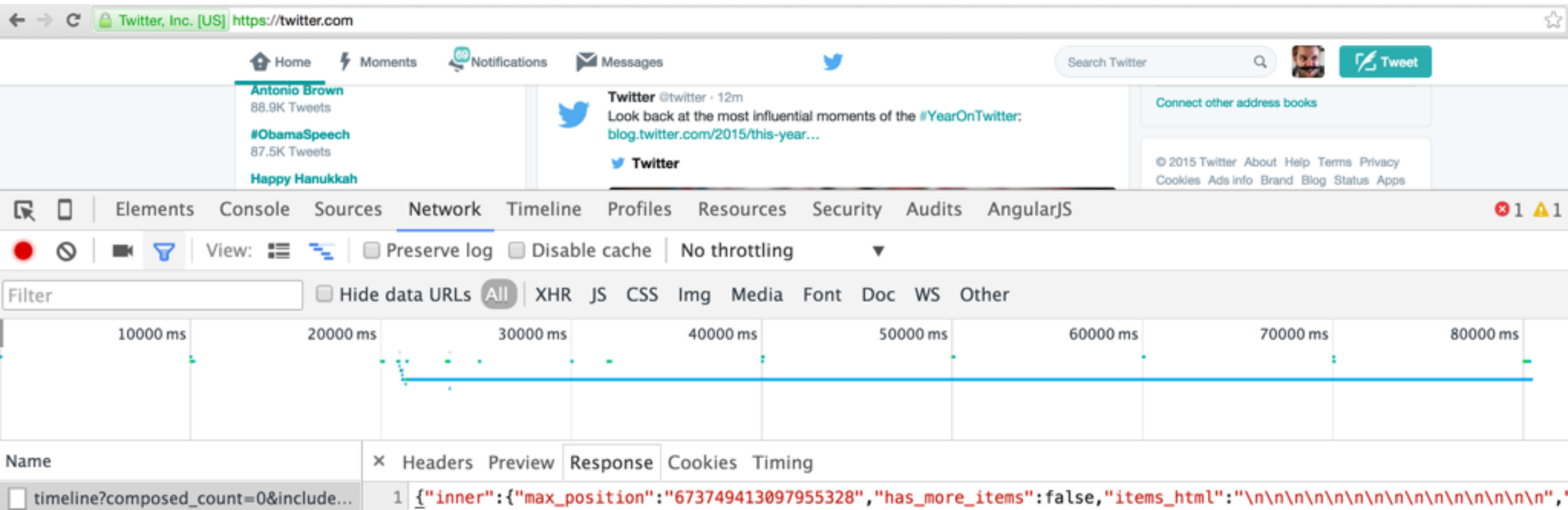
# JSON: WHY

➤ We want to do all this because…?

   ➤ Data lives over there

   ➤ We want it over here

   ➤ **Serialization** allows interchange

Request →

Response ←

Client

Server

# JSON: WHY



Web app → Request → App server
App server → Response → Web app

➤ When a web app needs an update,

➤ Web app requests new info from the server

➤ Gets a JSON response with data it can process

➤ Goes great with AJAX

# JSON: WHY

# JSON: WHY

➤ JSON is a standard way to **serialize** your objects

  ➤ "Dehydration for data structures"

  ➤ Reconstitute it later when needed

➤ Data can be persisted via the localStorage API

  ➤ Keep user preferences

    ➤ What tab was open?

    ➤ Which articles have I expanded?

  ➤ Cache data
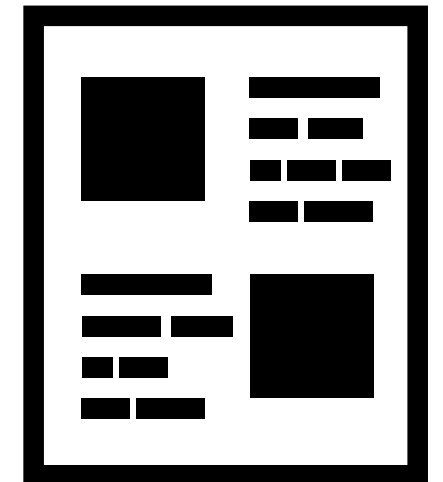
    ➤ Prevent unnecessary requests to the server

# JSON: BASIC RULES

➤ Always use double quotes

➤ Keys are quoted strings

➤ Values can be:

  ➤ String

  ➤ Number

  ➤ Object

  ➤ Array

  ➤ true/false

  ➤ null

# JSON

➤ Object Literal

```
1 {
2   author: 'Virginia Sawayn',
3   title: 'Navigating Solid State Multi-byte Monitors'
4 }
```
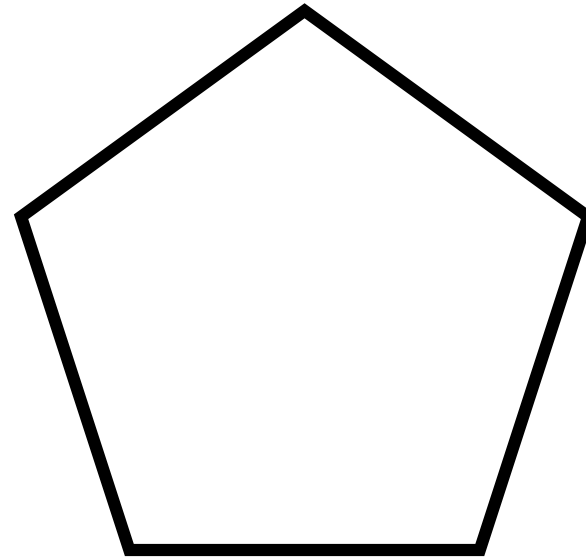
➤ JSON

```
1 {
2   "author": "Virginia Sawayn",
3   "title": "Navigating Solid State Multi-byte Monitors"
4 }
```

JSON: HOW

➤ Object Literal

```
1  {
2    name: 'Pentagon',
3    sides: 5,
4    sideLength: 12,
5    regular: true
6  }
```

➤ JSON

```
1  {
2     "name": "Pentagon",
3     "sides": 5,
4     "sideLength": 12,
5     "regular": true
6  }
```

JSON

➤ Object Literal

```
1  {
2    name: 'American Pharoah',
3    jockey: {
4      name: 'Victor Espinoza',
5      yob: 1972
6    },
7    breeder: {
8      name: 'Zayat Stables',
9      location: {
10       city: 'Hackensack',
11       state: 'New Jersey'
12     }
13   }
14 }
```

➤ JSON

```
1  {
2    "name": "American Pharoah",
3    "jockey": {
4        "name": "Victor Espinoza",
5        "yob": 1972
6    },
7    "breeder": {
8        "name": "Zayat Stables",
9        "location": {
10           "city": "Hackensack",
11           "state": "New Jersey"
12       }
13   }
14 }
```

JSON

```json
1 {
2   "pilots": [
3     {
4       "name": "Amelia Earhart",
5       "yob": 1897,
6       "yod": 1936
7     },
8     {
9       "name": "Anne Morrow Lindbergh",
10      "yob": 1906,
11      "yod": 2001
12    },
13    {
14      "name": "Chuck Yeager",
15      "yob": 1923,
16      "yod": null
17    }
18  ]
19 }
```

JSON: SUMMARY

➤ Lightweight data interchange

➤ Machines can talk to machines

➤ Our code can talk to output of other code

JSON: RESOURCES

➤ http://json.org/

➤ https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON

➤ http://www.drowningintechnicaldebt.com/RoyAshbrook/archive/2007/07/09/top-10-quot-why-xml-sucks-quot-articles.aspx

AJAX

AJAX: WHAT

➤ "Asynchronous JavaScript And XML"

➤ But's it's mostly JSON these days, as seen at <u>twitter.com</u>

➤ …or HTML that's retrieved on-demand

➤ No one wants to figure out how to pronounce…

    ➤ **AJAJ / AJAH**

AJAX: WHEN

➤ 1998: MS Outlook Web Access uses async with ActiveX

➤ Dec 6, 2000: Firefox releases JS compatible version

➤ 2004: Gmail, Support in Safari 1.2

➤ 2005: Google Maps, Google Suggest, "AJAX"

➤ 2006: Support in Internet Explorer 7

➤ Widespread usage followed

➤ "Web 2.0" was born!
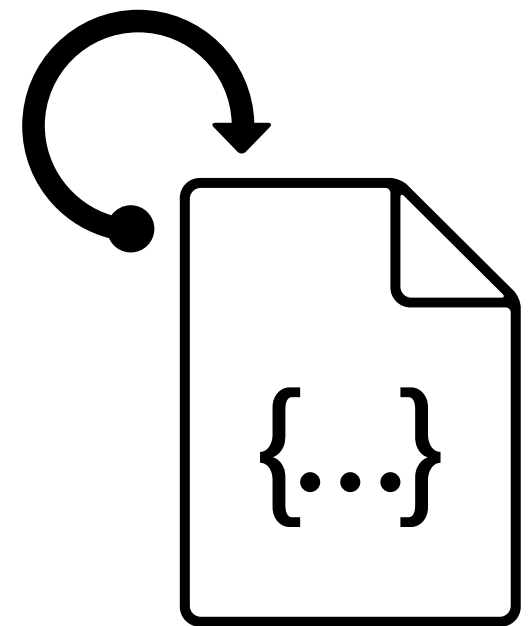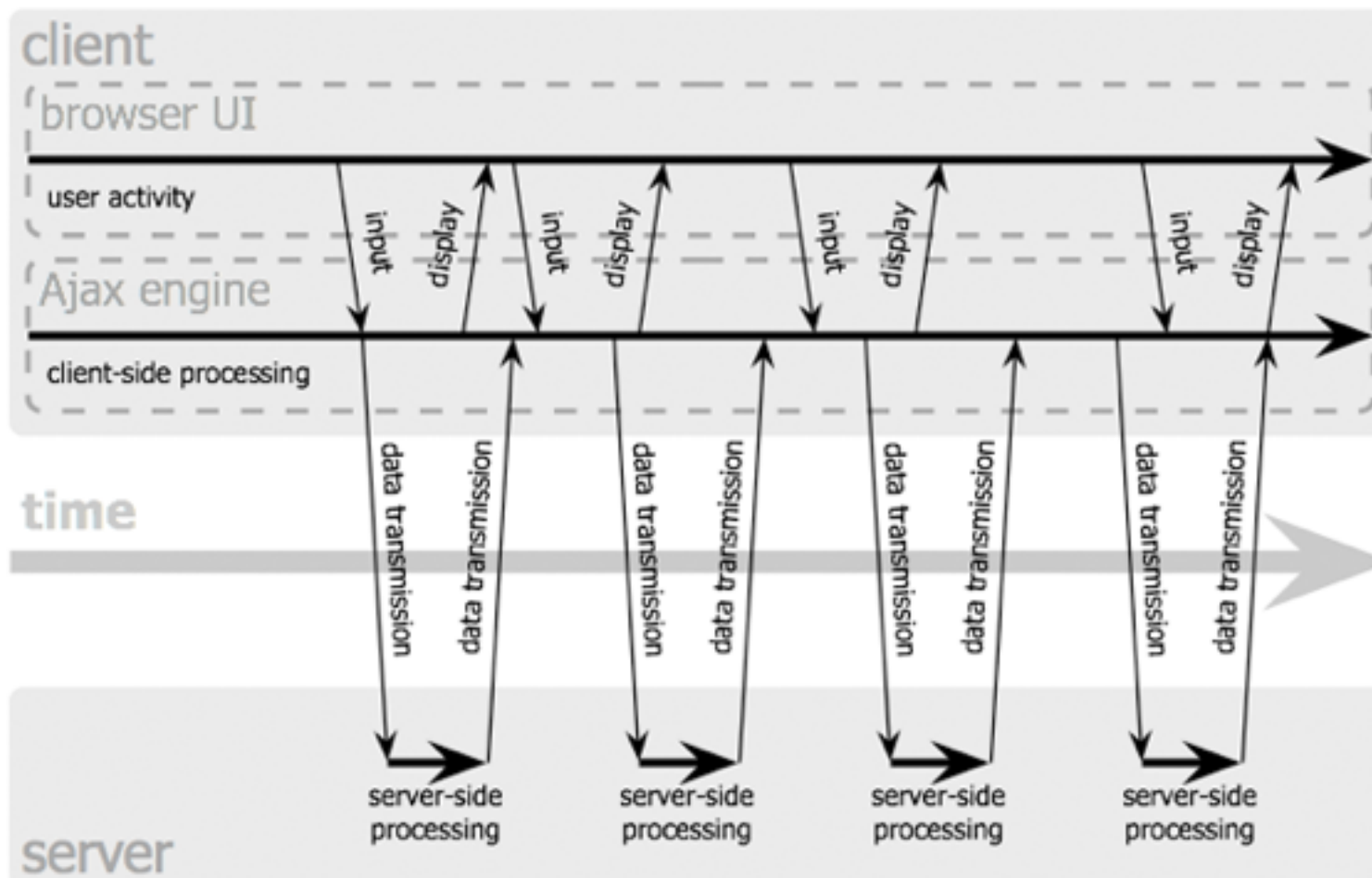
AJAX: WHAT

➤ Did you say **Asynchronous**?

➤ Async, like, with callbacks?

➤ AJAX is about asynchronous **requests**

   ➤ Don't reload the whole page & DOM

   ➤ Make a request programmatically

      ➤ No longer limited to script load order

      ➤ Could be during event handling

      ➤ Could be on a timer (see: twitter.com)

      ➤ Undetermined duration: Callback runs "when done"
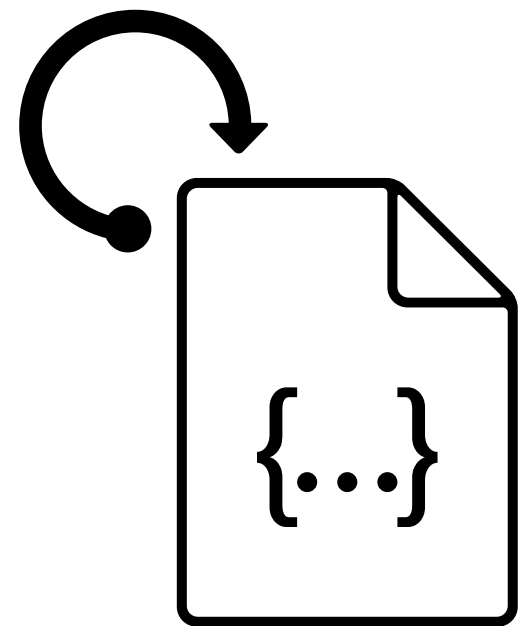
classic web application model (synchronous)
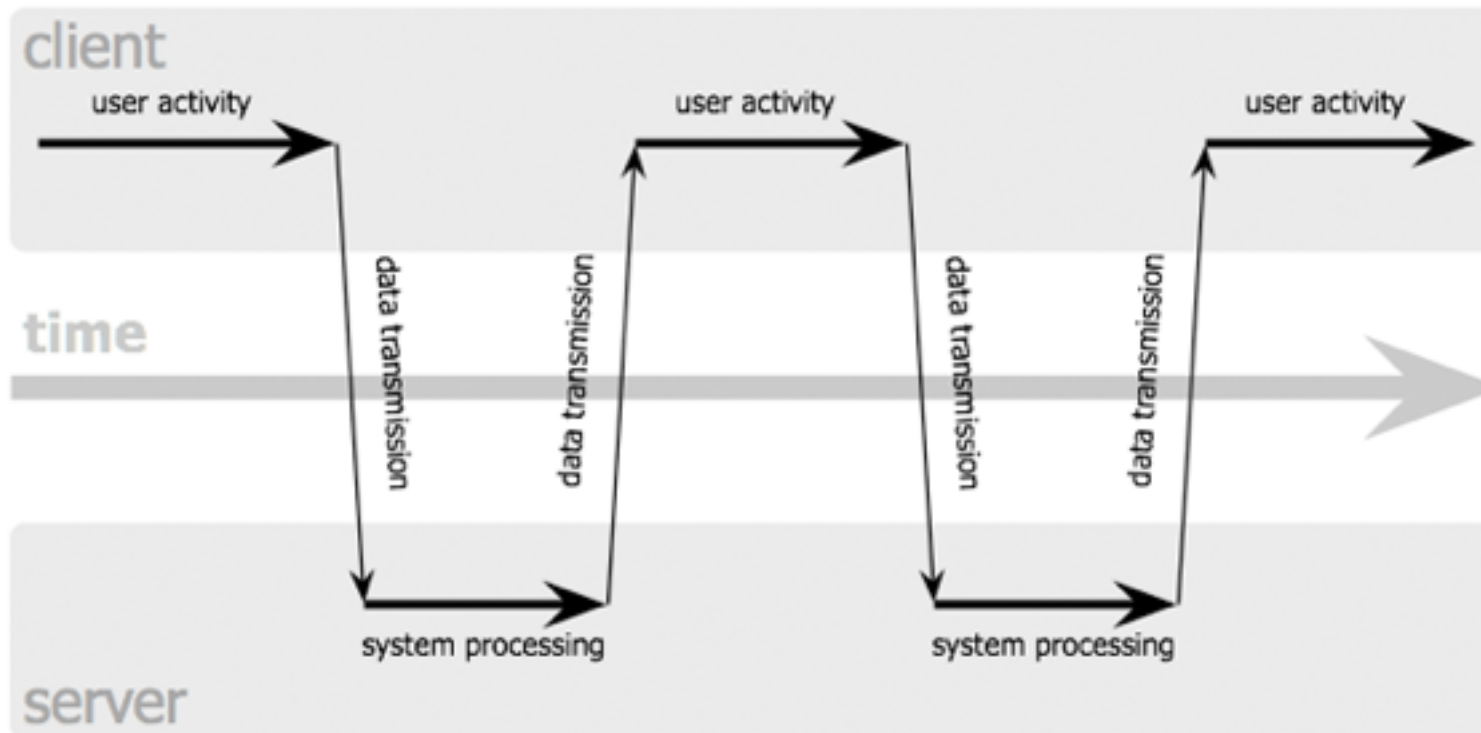
Ajax web application model (asynchronous)

Jesse James Garrett / adaptivepath.com

AJAX: POWER UP

➤ Enables an entirely new class of application

 ➤ Faster interactions mean more usable apps

 ➤ Fetch just what you need

 ➤ Update only what changes in the page

 ➤ Stay in touch with the server

AJAX: WITH JQUERY

1. Send a request

2. Register an async callback
   function to handle the response

3. Profit!

AJAX

Global Ajax Event Handlers
.ajaxComplete()
.ajaxError()
.ajaxSend()
.ajaxStart()
.ajaxStop()
.ajaxSuccess()

Helper Functions
jQuery.param()
.serialize()
.serializeArray()

Low-Level Interface
jQuery.ajax()
jQuery.ajaxSetup()

Shorthand Methods
jQuery.get()
jQuery.getJSON()
jQuery.getScript()
.load()
jQuery.post()

AJAX: SEND A REQUEST

1. Send a request

   ➤ Specify the request "method", URL, settings

   ➤ Request method options:

      ➤ GET

      ➤ POST

      ➤ HEAD

      ➤ etc…

   ➤ jQuery has helper methods to simplify:

      `$.ajax, $.load, $.get, $.getJSON, $.post`

AJAX: CALLBACKS

2. Register an async callback function to handle the response

- ➤ Write code to handle each kind of response:

  - ➤ `success` option or chain `.done()`

  - ➤ `error` option, or chain `.fail()`

  - ➤ complete option, or chain `.always()`

- ➤ Anything that depends on the response, must be in a callback!

- ➤ Use named functions to avoid callback-hell

  - ➤ Seriously: http://callbackhell.com/

AJAX DEMO

AJAX: SUMMARY

➤ Critically useful technique for making modern web apps

➤ Client can control communication with the server

AJAX: FURTHER RESOURCES

➤ https://en.wikipedia.org/wiki/XMLHttpRequest

➤ http://adaptivepath.org/ideas/ajax-new-approach-web-applications/

➤ https://teamtreehouse.com/library/ajax-basics

RECAP

RECAP

➤ JSON & AJAX: Taste great together

➤ Organize your data management

    ➤ Transit

    ➤ Storage & caching

REFERENCES AND SOURCES

➤ Vector icons via The Noun Project

➤ http://i.jeded.com/i/jason-and-the-argonauts-1963.29114.jpg

➤ https://commons.wikimedia.org/wiki/File:Amelia_Earhart_awaits_transatlantic_flight_1928.jpg

➤ https://en.wikipedia.org/wiki/Ajax_(programming)