

# JAVASCRIPT DEVELOPMENT

Sasha Vodnik, Instructor

# HELLO!

1. Pull changes from the `svodnik/jsd5` repo to your computer
2. Navigate to the `starter-code` folder

---

**JAVASCRIPT DEVELOPMENT**

---

# **AJAX & APIS**

# LEARNING OBJECTIVES

At the end of this class, you will be able to

- › Identify all the HTTP verbs & their uses.
- › Describe APIs and how to make calls and consume API data.
- › Access public APIs and get information back.
- › Implement an Ajax request with vanilla JS.
- › Implement a jQuery Ajax client for a simple REST service.
- › Reiterate the benefits of separation of concerns – API vs. Client.

# **AGENDA**

- APIs
- HTTP
- Ajax & JavaScript
- Ajax & jQuery

# THE DOM, JQUERY, & TEMPLATING — REVIEW

- In the channel for today (08-dom-jquery-cont), share your answers to one or both of the following questions:
  - **Significant thing:**  
“The most significant thing I learned about the DOM, jQuery, and templating is \_\_\_\_\_.”
  - **Outstanding question:**  
“My biggest outstanding question on the DOM, jQuery, or templating is \_\_\_\_\_.”

# Checkin and questions

- The **most significant thing I learned** about the DOM, jQuery, and templating is \_\_\_\_\_.
- My **biggest outstanding question** about the DOM, jQuery, and templating is \_\_\_\_\_.

# **What kinds of data are available online?**



# APIs

# WEB SERVICE

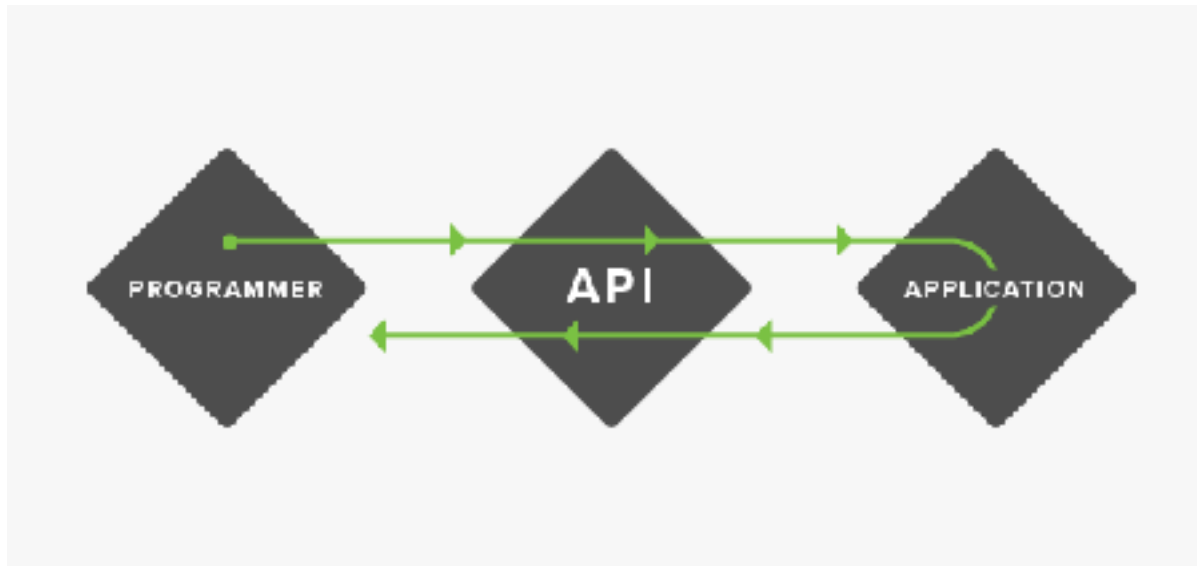
- An online source of data
- Communicate using HTTP, but instead of markup we receive data
- We can use multiple services in a single app
- This makes separation of concerns (DOM logic and data) even more important

# API = application programming interface

- Each service has an API, which is a predefined set of objects, properties, and methods anyone can use to access that service
- Any service we access online through our apps will have an API
- Middleman; allows different pieces of software to communicate

# APIS IN THE REAL WORLD

- Most APIs are unique, like separate languages
- API for devices (iPhone); for operating systems (macOS); for JavaScript libraries (jQuery API)



# HOW WE WILL USE APIS

- We will focus on web-based APIs (for web services)
- Use HTTP to request/receive structured data from endpoints on a server
- Endpoints are addresses (URLs) that will return data (JSON) instead of markup (HTML)

# WHAT WE NEED TO KNOW TO USE AN API

- Its terms of service (paid service? limit on usage?)
- How to make a request (URL and parameters)
- What kind of data is returned and how to parse it

# HOW MIGHT A SERVICE REQUEST BE DIFFERENT THAN USING OUR OWN DATA?

- May need to authenticate when requesting data
- May be a lag, requiring user notification
- Request may result in an error

# **REST (representational state transfer)**

- architectural style of web applications
- transfers a representation of the state of a server resource to the client
- [https://en.wikipedia.org/wiki/Representational\\_state\\_transfer](https://en.wikipedia.org/wiki/Representational_state_transfer)



# **RESTful API**

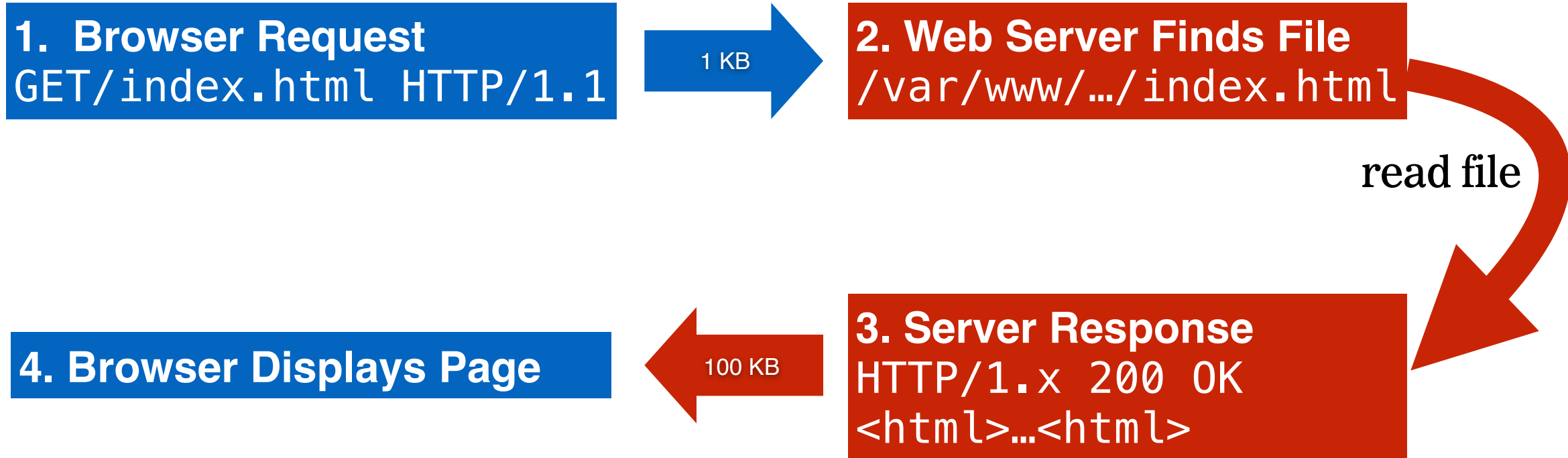
- adheres to REST architecture
- uses
  - a base URL
  - an Internet media type (such as JSON)
  - standard HTTP methods

# HTTP

# HTTP (hypertext transfer protocol)

- System of rules for how web pages are transmitted between computers
- Defines the format of messages passed between HTTP clients and HTTP servers
- A client sends a **request** to a server.
- A server sends a **response** back to a client.

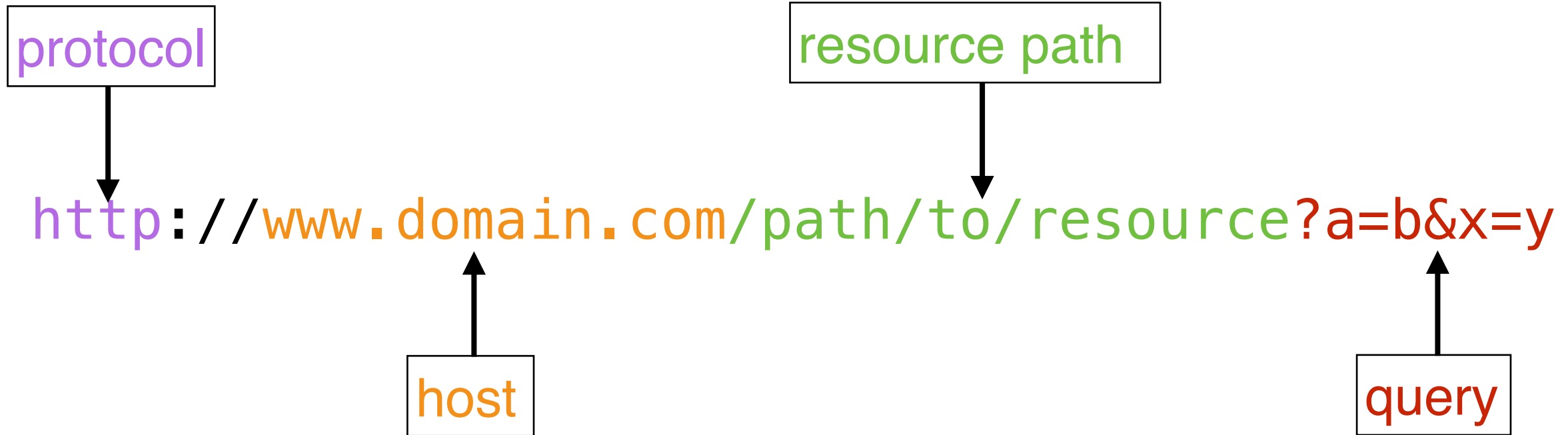
# HTTP REQUEST AND RESPONSE



# HTTP (hypertext transfer protocol)

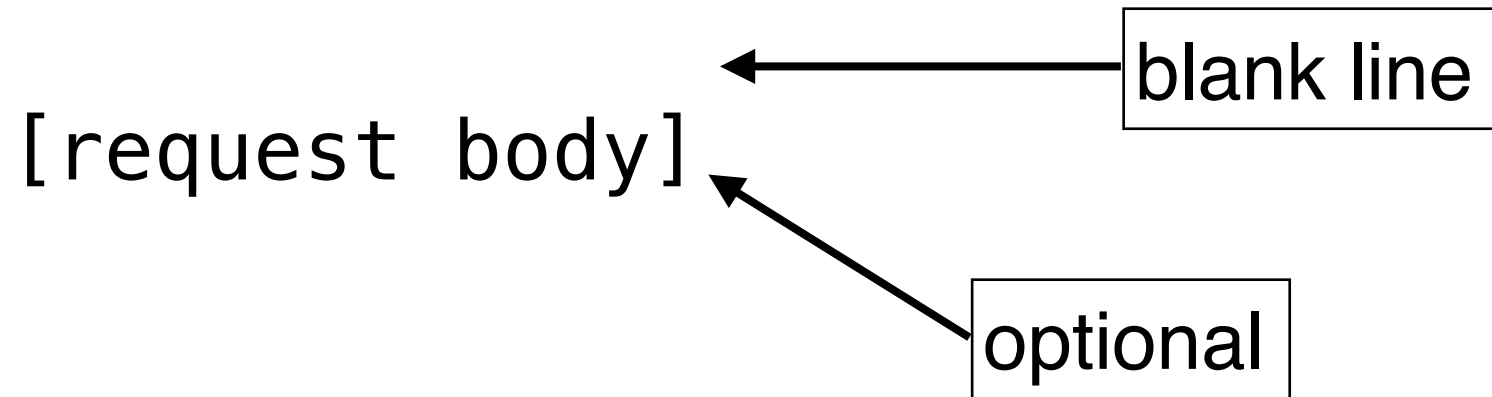
- HTTP clients are generally web browsers (Chrome, Firefox, Safari, Edge, etc.)
- HTTP servers are web servers (Apache, Nginx, etc.)
- **Web applications** are programs that plug into a web server, process the HTTP requests that the server receives, and generate HTTP responses

# HTTP REQUESTS IN EVERYDAY LIFE



# HTTP REQUEST STRUCTURE

[http request method] [URL] [http version]  
[list of headers]



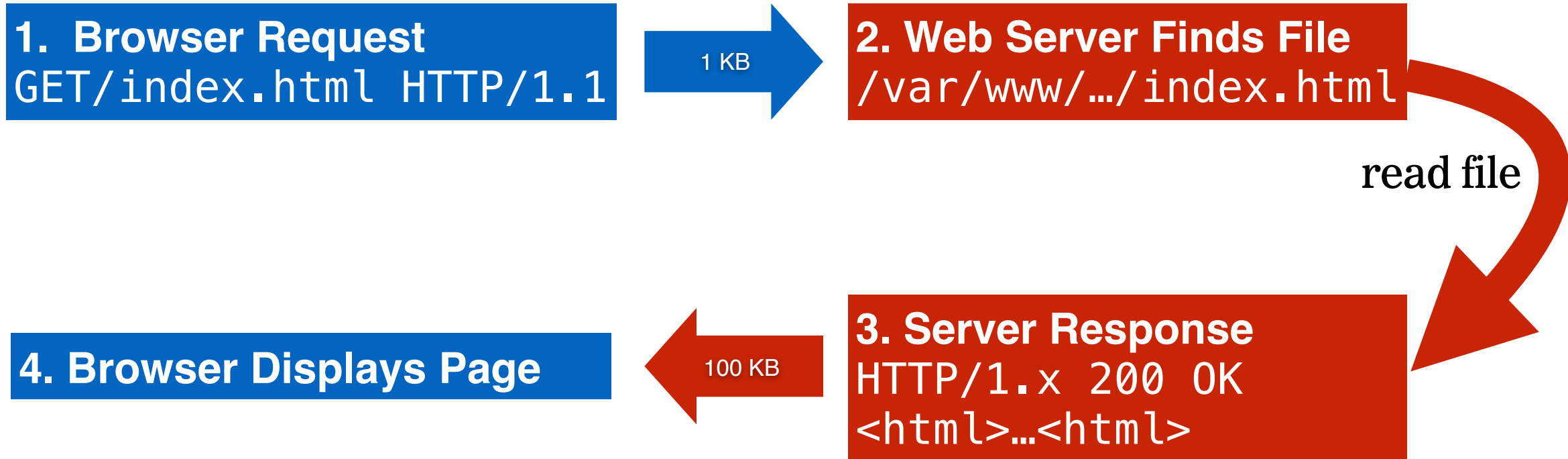
# HTTP REQUEST METHODS (“HTTP VERBS”)

GET	Retrieve a resource
POST	Create a resource
PATCH	Update an existing resource (same as PUT, but PATCH is recommended)
DELETE	Delete a resource
HEAD	Retrieve the headers for a resource

Most widely used



# HTTP REQUEST AND RESPONSE



# HTTP RESPONSE STRUCTURE

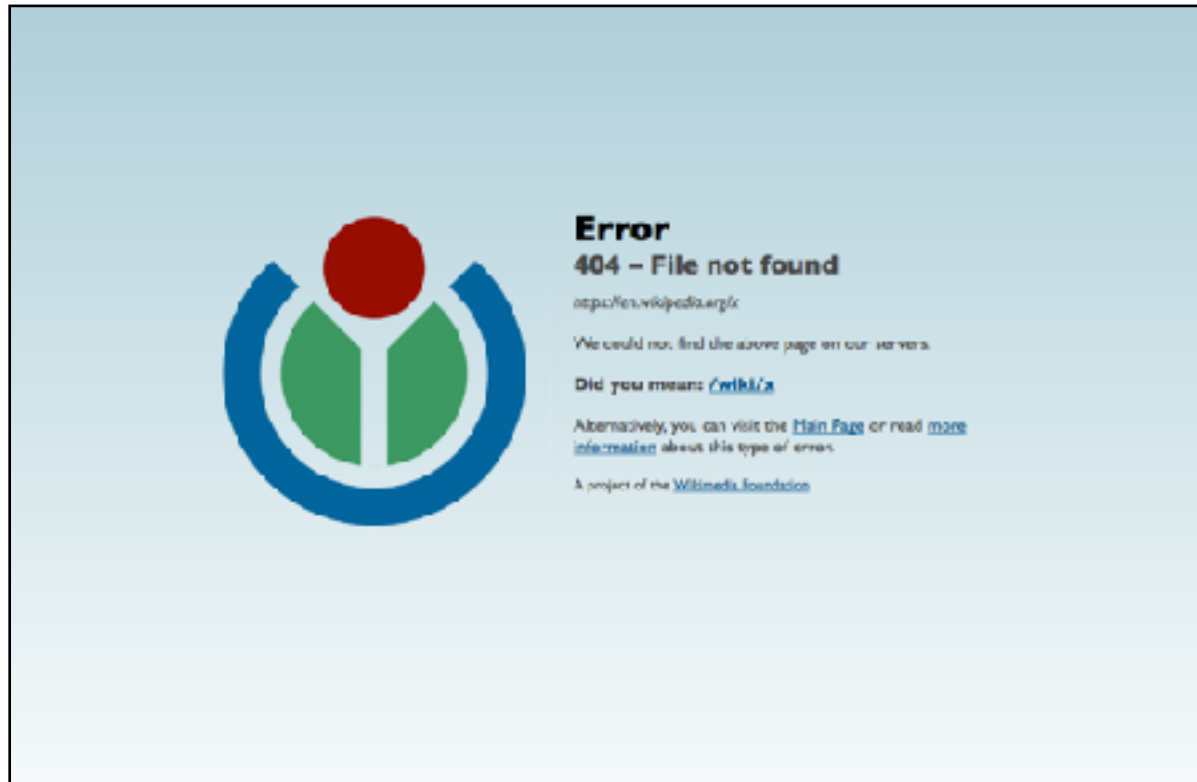
[http version] [status] [reason]  
[list of headers]

← blank line

[response body]

↙ usually HTML, JSON, etc

# HTTP STATUS CODES



# HTTP STATUS CODES

200	Okay
301	Moved permanently
302	Moved temporarily
400	Bad request
403	Forbidden
404	Not found
500	Internal server error

# **BREAK (5 MINUTES)**

# Ajax

# Ajax

- Originally AJAX (Asynchronous JavaScript and XML)
- XML is a format for data interchange that's derived from the same markup language that HTML comes from.
- Since JSON was codified, it's become the standard for data interchange on the web, so Ajax is no longer functionally an acronym.

# **What does Ajax let us do?**

- Communicate with servers from within our apps
- Make the communication asynchronous (in the background)
- We can update interfaces and content without refreshing the page



# XMLHttpRequest

- Standard object that we create an instance of for an HTTP request

method	description
onreadystatechange	we assign a custom function that specifies how we want to handle the HTTP response
open	opens HTTP connection; we specify request type and URL as parameters
send	sends request; generally no parameters needed

# XMLHttpRequest.readyState

- Property that stores the state of the request

value	state	description
0	UNSENT	Client has been created. <code>open()</code> not called yet.
1	OPENED	<code>open()</code> has been called.
2	HEADERS_RECEIVED	<code>send()</code> has been called, and headers and status are available.
3	LOADING	downloading; <code>responseText</code> holds partial data.
4	DONE	The operation is complete.

# **BREAK (5 MINUTES)**

# **jQuery Ajax**

# Using Ajax with jQuery

method	description
<code>\$.get()</code>	loads data from a server using an HTTP GET request
<code>\$.ajax()</code>	performs an Ajax request based on parameters you specify

# LEARNING OBJECTIVES – REVIEW

- Identify all the HTTP Verbs & their uses.
- Describe APIs and how to make calls and consume API data.
- Access public APIs and get information back.
- Implement an Ajax request with vanilla JS.
- Implement a jQuery Ajax client for a simple REST service.
- Reiterate the benefits of separation of concerns – API vs. Client.

## **NEXT CLASS PREVIEW**

### **Asynchronous JavaScript and Callbacks**

- Store and use anonymous functions in variables.
- Pass functions as arguments to functions that expect them.
- Write functions that take other functions as arguments.
- Return functions from functions.

# Exit Tickets!



# Q&A