

# JAVASCRIPT DEVELOPMENT

*Sasha Vodnik, Instructor*

# HELLO!

1. Pull changes from the `svodnik/JS-SF-14-resources` repo to your computer
2. Open the `09-ajax-apis/starter-code` folder in your code editor

---

**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 Fetch.
- Create an Ajax request using jQuery.

# **AGENDA**

- APIs & HTTP
- Ajax using Fetch
- Ajax & jQuery
- Separation of concerns

---

## AJAX & APIS

---

# WEEKLY OVERVIEW

**WEEK 6**

Ajax & APIs / Asynchronous JS & callbacks

**WEEK 7**

Advanced APIs / Project 2 lab

**WEEK 8**

Prototypal inheritance / Closures & this

# **EXIT TICKET QUESTIONS**

1. How to use Template Literals?
2. How to get event delegation working

---

**AJAX & APIS**

---

# **HOMework REvIEW**

---

# HOMEWORK — GROUP DISCUSSION

---



EXERCISE

## **TYPE OF EXERCISE**

---

▸ Pairs

## **TIMING**

---

*4 min*

1. Share your solutions for the homework.
2. Share one thing you found challenging. If you worked it out, share how; if not, brainstorm with your group how you might approach it.

---

**ADVANCED JQUERY**

---

# **METHOD CHAINING**

# CHAINING

without chaining:

```
let $mainCaption = $('<p>');  
let $captionWithText = $mainCaption.html('Today');  
let $fullCaption = $captionWithText.addClass('accent');
```

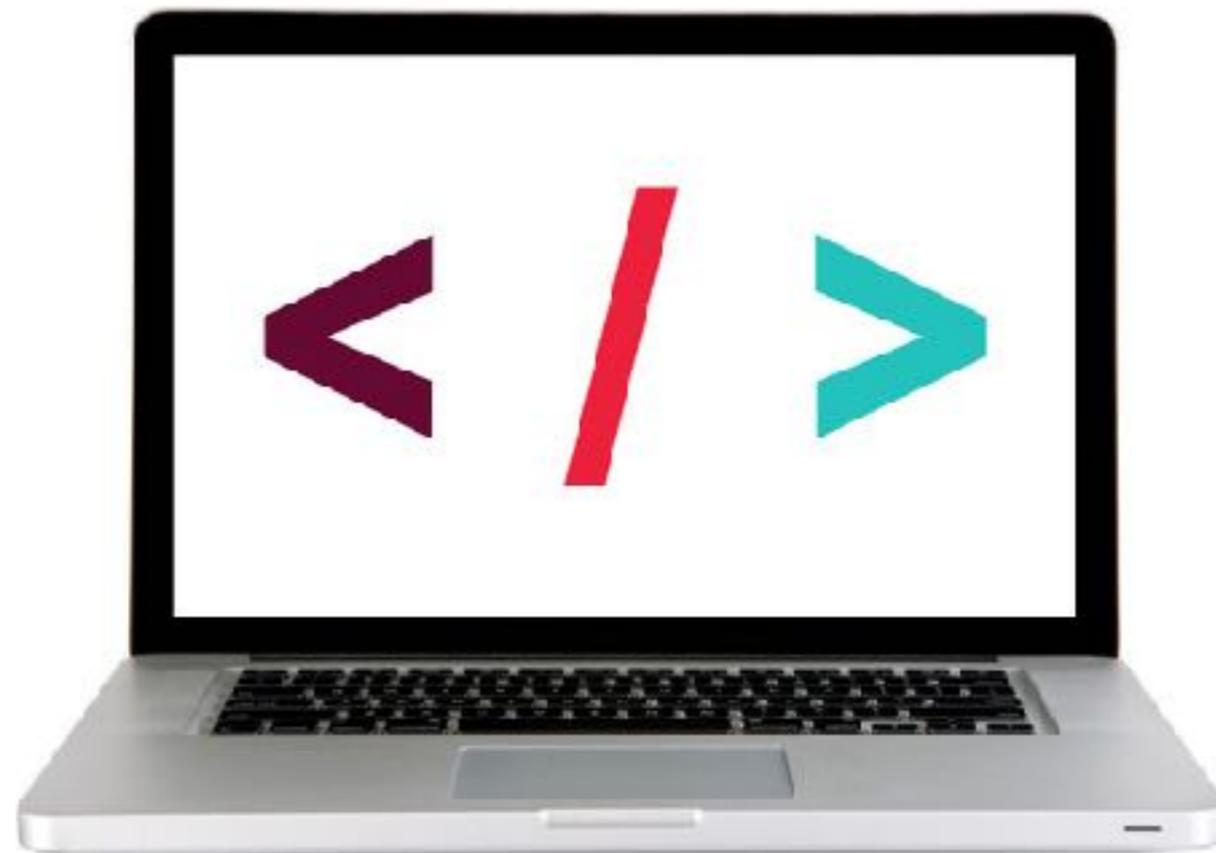
with chaining:

```
let $fullCaption = $('<p>').html('Today').addClass('accent');
```

---

## ADVANCED JQUERY

---



**LET'S TAKE A CLOSER LOOK**

---

# EXERCISE - CHAINING

---



EXERCISE

## **OBJECTIVE**

---

- ▶ Use chaining to place methods on selectors.

## **LOCATION**

---

- ▶ `starter-code > 10-best-practices-exercise`

## **TIMING**

---

*3 min*

1. In your browser, open `index.html` and test the functionality.
2. Open `main.js` in your editor and complete items 1 and 2.
3. In your browser, reload `index.html` and verify that the functionality is unchanged.

---

**ADVANCED JQUERY**

---

# IMPLICIT ITERATION

# IMPLICIT ITERATION

## explicit iteration

```
$('.li').each(function() {  
  $(this).removeClass('current');  
});
```

jQuery .each() method works like a forEach loop



not necessary for  
element collections

## implicit iteration

```
$('.li').removeClass('current');
```

applying any method to a jQuery collection iterates through each element!

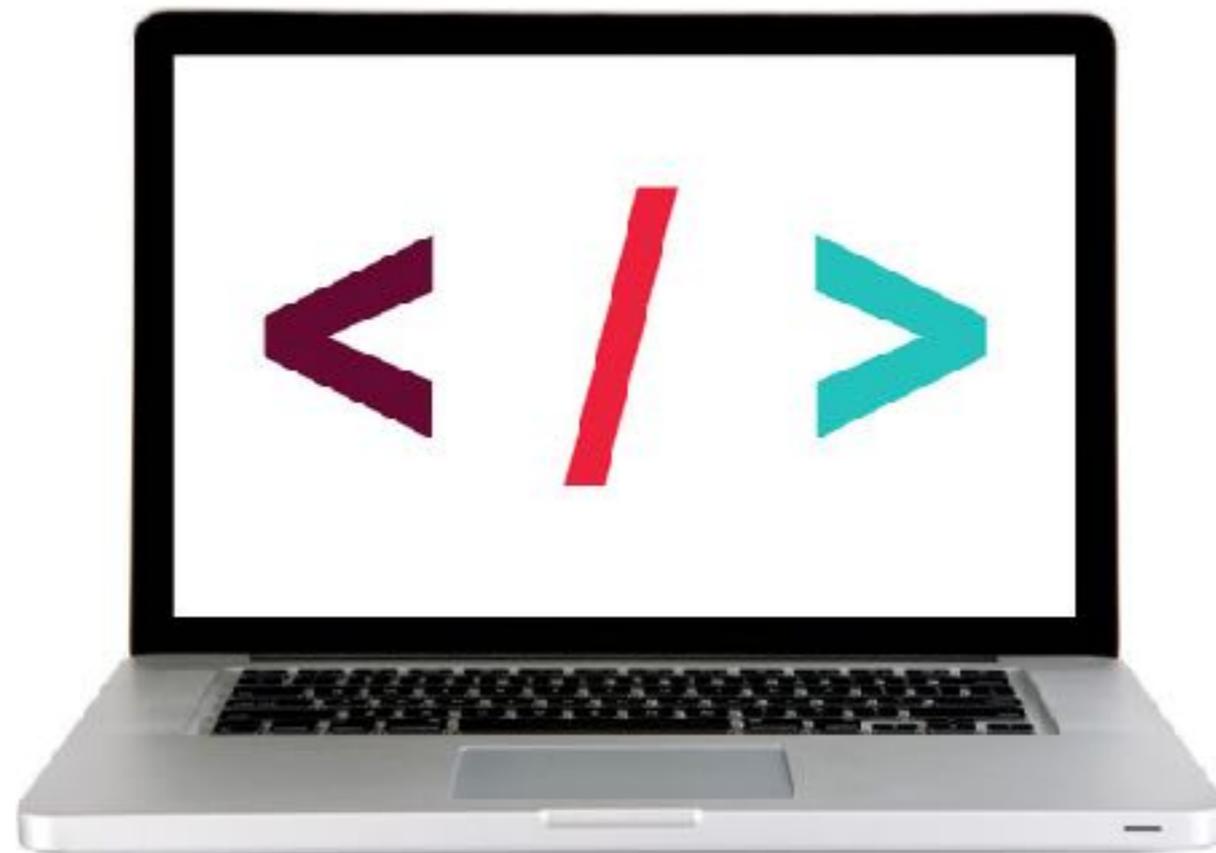


less code = best practice!

---

## ADVANCED JQUERY

---



**LET'S TAKE A CLOSER LOOK**

---

# EXERCISE – IMPLICIT ITERATION

---



## **OBJECTIVE**

---

- ▶ Use implicit iteration to update elements of a jQuery selection.

## **LOCATION**

---

- ▶ `starter-code > 10-best-practices-exercise`

## **TIMING**

---

*5 min*

1. Return to `main.js` in your editor and complete item 3.
2. In your browser, reload `index.html` and verify that the functionality is unchanged.

---

**ADVANCED JQUERY**

---

# EVENT DELEGATION

# WITHOUT EVENT DELEGATION

1. load page

2. set event listener on list items

3. add a new list item

```
$( 'li' ).on( 'click', function() {
  addClass( 'selected' )
});
```

- item1
- item2
- item3

- item1 click event
- item2 click event
- item3 click event

- item1 click event
- item2 click event
- item3 click event
- item4

click event is not automatically applied to the new li element



# WITH EVENT DELEGATION

1. load page

- item1
- item2
- item3

2. set event listener  
on *parent* of list items

selector  
changed to  
parent

new second  
argument  
specifies children

```
$('ul').on('click', 'li', function(){  
  addClass('selected')  
});
```

- item1 click event
- item2 click event
- item3 click event

3. add a new list item

- item1 click event
- item2 click event
- item3 click event
- item4 click event

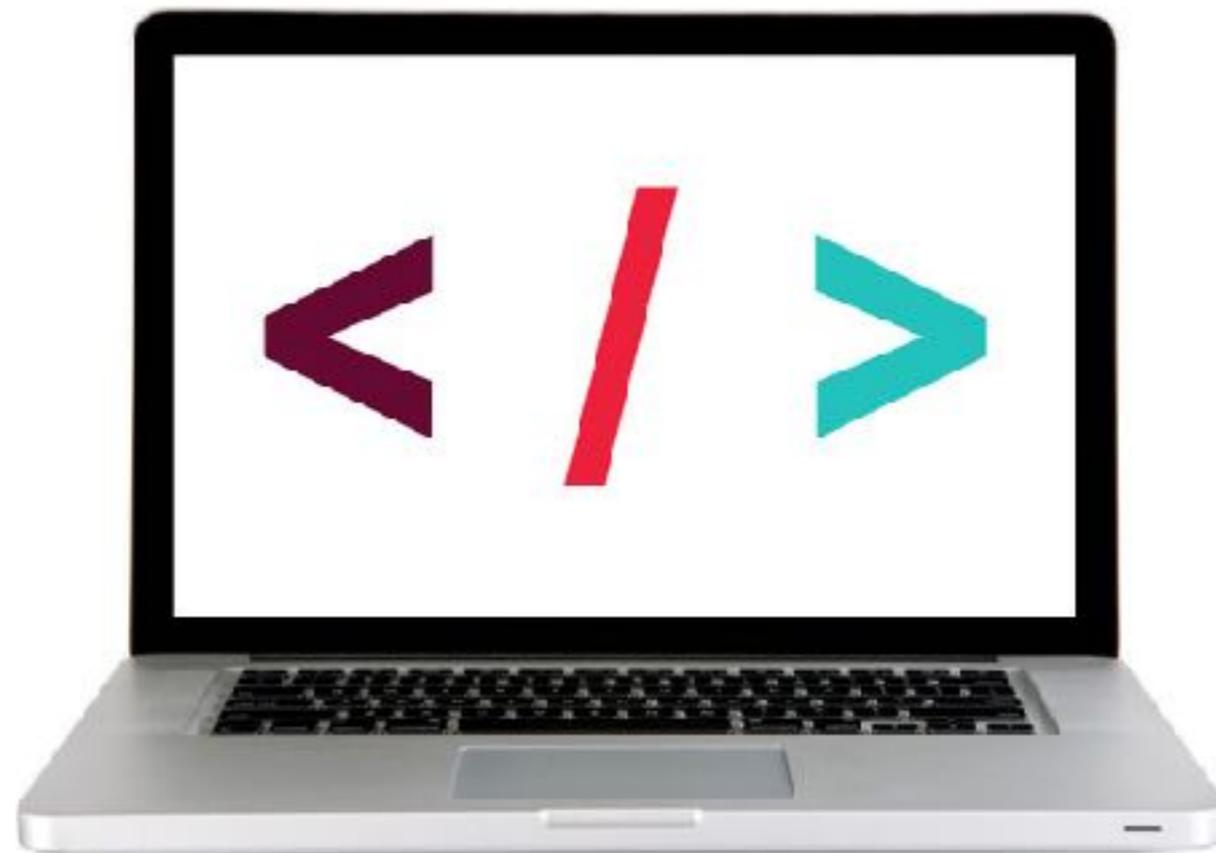
click event IS automatically applied to the new li element!



---

## ADVANCED JQUERY

---



**LET'S TAKE A CLOSER LOOK**

---

# EXERCISE – EVENT DELEGATION

---



EXERCISE

## **OBJECTIVE**

---

- ▶ Use event delegation to manage dynamic content.

## **LOCATION**

---

- ▶ `starter-code > 10-best-practices-exercise`

## **TIMING**

---

*10 min*

1. Return to `main.js` in your editor and complete item 4.
2. In your browser, reload `index.html` and verify that when you add a new item to the list, its “cross off” link works.
3. BONUS 1: When the user mouses over each item, the item should turn grey. Don't use CSS hovering for this.
4. BONUS 2: Add another link, after each item, that allows you to delete the item.

# **ATTACHING MULTIPLE EVENTS WITH A SINGLE ON() STATEMENT**

# ATTACHING MULTIPLE EVENTS WITH A SINGLE .ON() STATEMENT

- ▶ We could write a separate .on() statement for each event on an element:

```
var $listElement = $('#contents-list');

$listElement.on('mouseenter', 'li', function(event) {
    $(this).siblings().removeClass('active');
    $(this).addClass('active');
});

$listElement.on('mouseleave', 'li', function(event) {
    $(this).removeClass('active');
});
```

# ATTACHING MULTIPLE EVENTS WITH A SINGLE .ON() STATEMENT

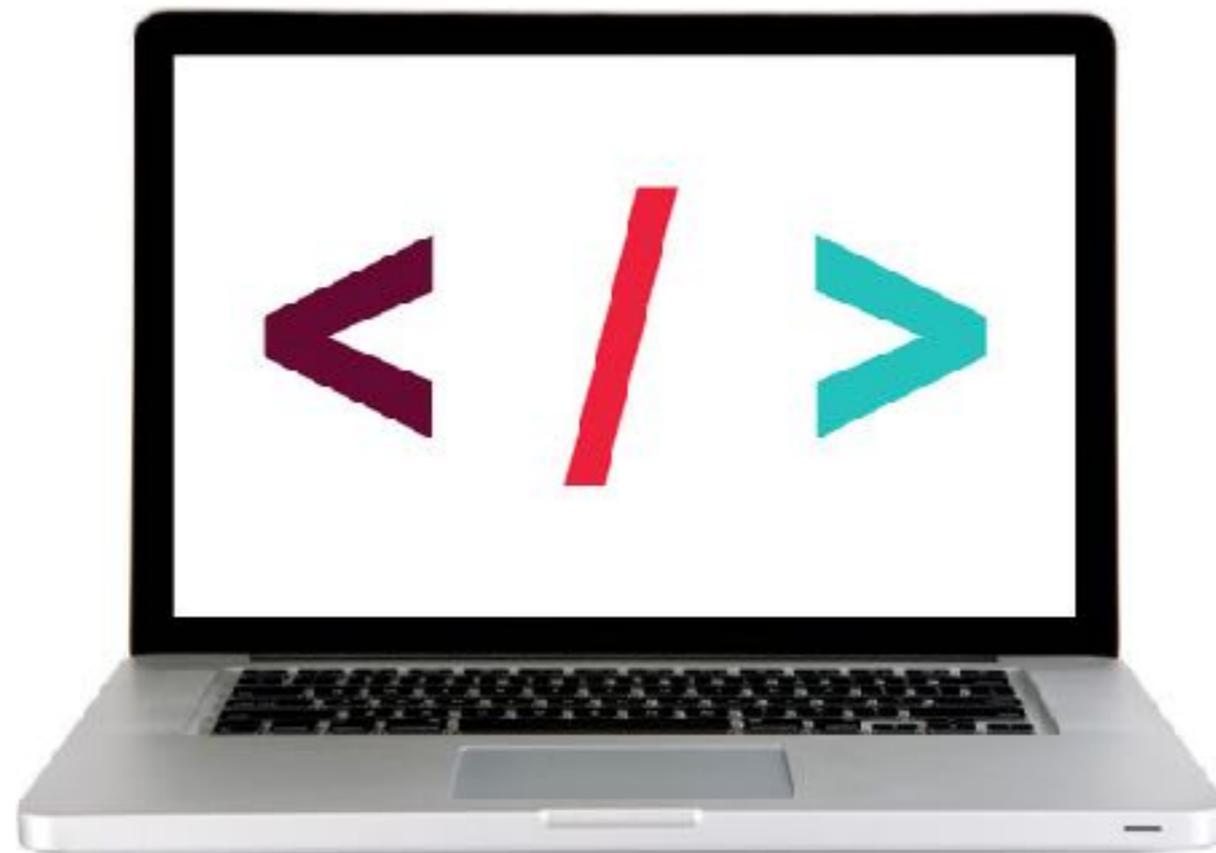
```
var $listElement = $('#contents-list');

$listElement.on('mouseenter mouseleave', 'li', function(event) {
    if (event.type === 'mouseenter') {
        $(this).siblings().removeClass('active');
        $(this).addClass('active');
    } else if (event.type === 'mouseleave') {
        $(this).removeClass('active');
    }
});
```

---

## ADVANCED JQUERY

---



**LET'S TAKE A CLOSER LOOK**

---

# EXERCISE - ATTACHING MULTIPLE EVENTS

---



EXERCISE

## **LOCATION**

---

▶ starter-code > 11-multiple-events-exercise

## **TIMING**

---

*5 min*

1. In your browser, open `index.html`. Move the mouse over each list item and verify that the sibling items turn gray.
2. In your editor, open `main.js` and refactor the two event listeners near the bottom of the file into a single event listener for multiple events.
3. In your browser, reload `index.html` and verify that the functionality is unchanged.

# LAB - JQUERY HOMEWORK

---



EXERCISE

## **LOCATION**

---

▶ 08-advanced-jquery > starter-code > Homework-4 > survey-form

## **TIMING**

---

*10 min*

1. Now that you have a clearer understanding of event delegation, implement this feature in the survey-form exercise from last week's homework.
2. When you complete the project (or if you already have completed it), please make yourself available to classmates who have questions.

---

---

# AJAX & APIS

---

# ACTIVITY

---



## **TYPE OF EXERCISE**

---

▶ Individual/Partner

## **TIMING**

---

*3 min*

1. Think about how you could use one or more sources of web data in an app.
2. Write a description or sketch a schematic of your app on your desk.

# APIs

# WEB SERVICES

Your app



Web service



request for data



response containing data



your app can now incorporate data from the web service

my website content



# SASHA VODNIK

Instructor and Author on Programming and Technology

Home

Books

impersonating you and resetting your password to one they choose. The result is that they have access to your account, while you are locked out. To defend against this type of attack, many web services allow you to set up two factor authentication (2FA).

Continue reading →

Share this:



★ Like 

One blogger likes this.

Content from Twitter added using Twitter API



## FOLLOW ME ON TWITTER

Tweets by @sashvodnik



Sasha Vodnik @sashvodnik

Take the next step to secure your passwords, browsing, and networking at my workshop this Thursday! [@GA\\_SF](#) [@GA](#) [#onlineSecurity](#) [#antioSecurity](#) [generalassembly](#) [byed.academy](#)...



Securing Your Digital ...  
Learn to keep your data ...  
generalassembly



Mar 4, 2018



Sasha Vodnik @sashvodnik

AR makes so much more sense to me than what VR, and here are some reasons why! ["LukoVR's Augmented Reality Examples"](#) by [@jeremiaszwhar.com](#) [/moments/6677](#) ...



LukoVR's Augmented Reality Examples

by [Jeremy Q.](#) ...

## Securing Your Digital Life, Part 1: Choosing a password manager

JANUARY 11, 2018

Configuring and using a password manager is a critical building block of your online security.

Continue reading →

Share this:



★ Like 

One blogger likes this.

Kayak  
website  
content



KAYAK Hotels Flights Cars Packages More

One-way 2 travelers Economy

Dublin (DUB) San Francisco (SFO) Thu 8/23

Our Advice Buy now

Prices are unlikely to decrease within 7 days

Track Prices

887 of 1411 flights Sorted by Recommended

Fee Assistant Carry-on bag 0 Checked bag 0

Orbitz Flight Deals - Price Guarantee

Plus earn Orbitz Rewards

Orbitz.com | Sponsored

View Deal

BEST FLIGHTS

8/24 Fri	Multiple Airlines	11:45 am DUB	4:16 pm OAK (+1)	8h 30m	\$360 KIWI.COM	View Deal
8/26 Sun	WOW air	11:45 am DUB	5:15 pm SFO	13h 30m	\$419 KAYAK	View Deal
8/23 Thu	Aer Lingus	12:30 pm DUB	3:30 pm SFO	11h 00m	\$851 OneTravel	View Deal

\$856 book early on KAYAK

Content  
from  
kiwi.com  
using API

Content  
from  
OneTravel  
using API

# WEB SERVICES



OMDb



# API = application programming interface



## By city ID

### Description:

You can call by city ID. API responds with exact result.

List of city ID city.list.json.gz can be downloaded here <http://bulk.openweathermap.org/sample/>

We recommend to call API by city ID to get unambiguous result for your city.

### Parameters:

id City ID

### Examples of API calls:

[api.openweathermap.org/data/2.5/weather?id=2172797](http://api.openweathermap.org/data/2.5/weather?id=2172797)

## By geographic coordinates

### API call:

[api.openweathermap.org/data/2.5/weather?lat={lat}&lon={lon}](http://api.openweathermap.org/data/2.5/weather?lat={lat}&lon={lon})

### Parameters:

# APIS IN THE REAL WORLD

▸ Most APIs are unique, like separate languages

▸ APIs for

▸ devices (iPhone)



▸ operating systems (macOS)



Mac OS



▸ JavaScript libraries (jQuery API)



▸ services (Slack)



# WEB SERVICES



OMDb



## ENDPOINTS

▶ Addresses (URLs) that return data (JSON) instead of markup (HTML)

### By city ID

Description:

You can call by city ID. API responds with exact result.

List of city ID `city.list.json.gz` can be downloaded here <http://bulk.openweathermap.org/sample/>

We recommend to call API by city ID to get unambiguous result for your city.

Parameters:

id City ID

Examples of API calls:

`api.openweathermap.org/data/2.5/weather?id=2172797`

### By geographic coordinates

API call:

`api.openweathermap.org/data/2.5/weather?lat={lat}&lon={lon}`

Parameters:

lat, lon coordinates of the location of your interest

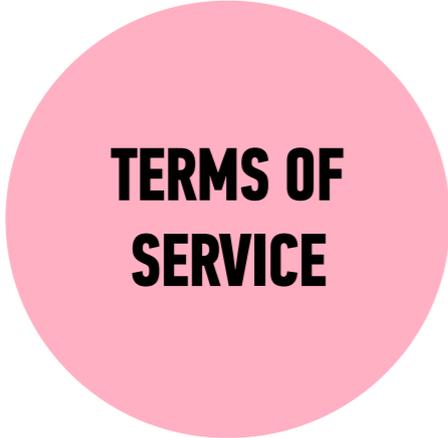
Examples of API calls:

`api.openweathermap.org/data/2.5/weather?lat=35&lon=139`

API respond:

```
[{"coord": {"lon": 139, "lat": 35},
"sys": {"country": "JP", "sunrise": 1369769524, "sunset": 1369821849},
"weather": [{"id": 804, "main": "clouds", "description": "overcast clouds", "icon": "04n"}],
"main": {"temp": 289.5, "humidity": 89, "pressure": 1013, "temp_min": 287.04, "temp_max": 292.04},
"wind": {"speed": 7.31, "deg": 187.002},
"rain": {"3h": 0},
"clouds": {"all": 92},
"dt": 1369824608}
```

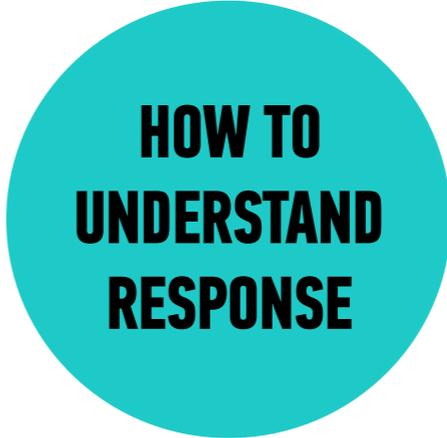
# **WHAT WE NEED TO KNOW TO USE AN API**



**TERMS OF  
SERVICE**

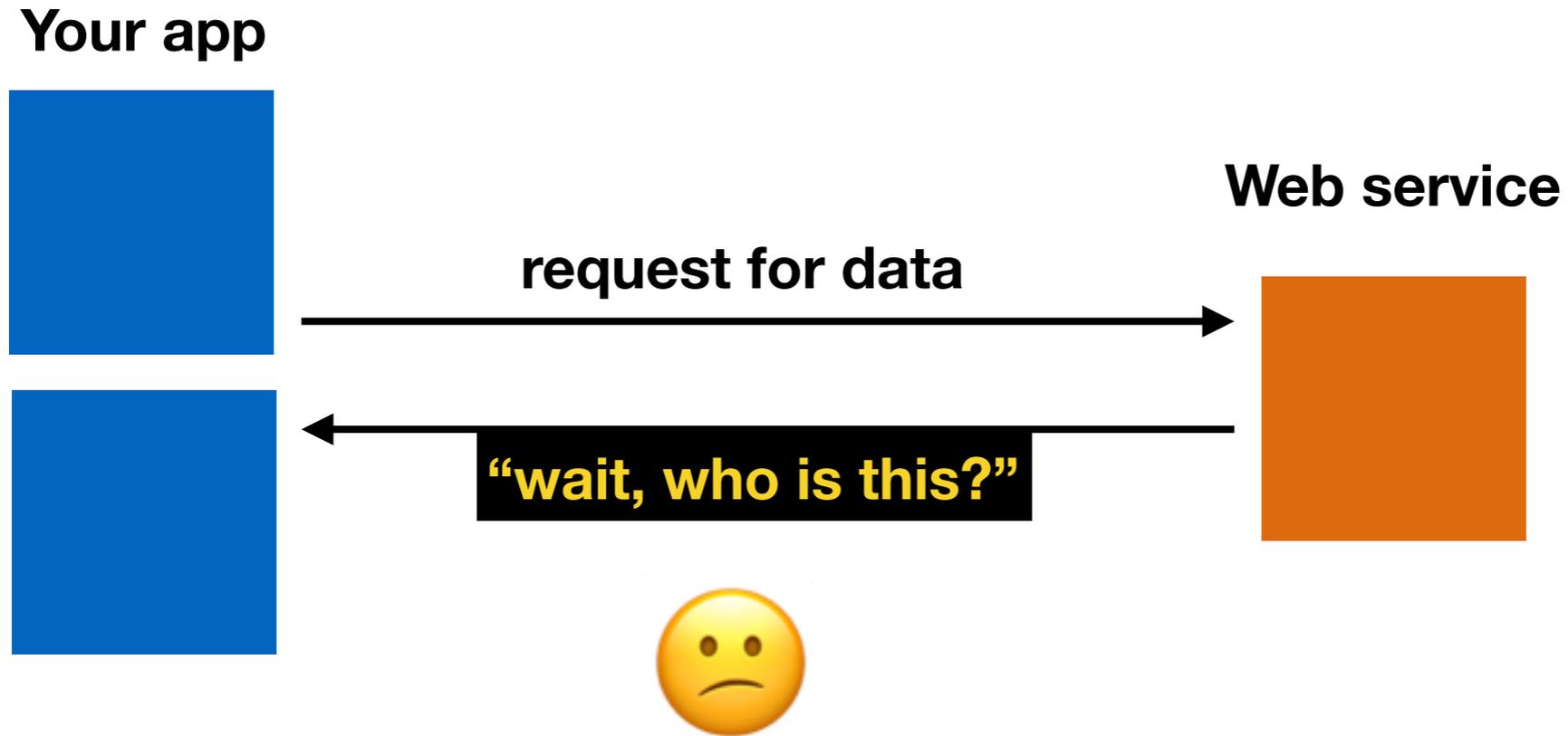


**HOW TO  
MAKE A  
REQUEST**



**HOW TO  
UNDERSTAND  
RESPONSE**

# AN API MIGHT REQUIRE AUTHENTICATION



# API KEY



asd32f;li^6uq439#0587adf;lgk@

- unguessable character string
- connects your requests to your account

# API REQUEST WITH AUTHENTICATION

Your app



request for data  
+ API key



Web service



authorization successful

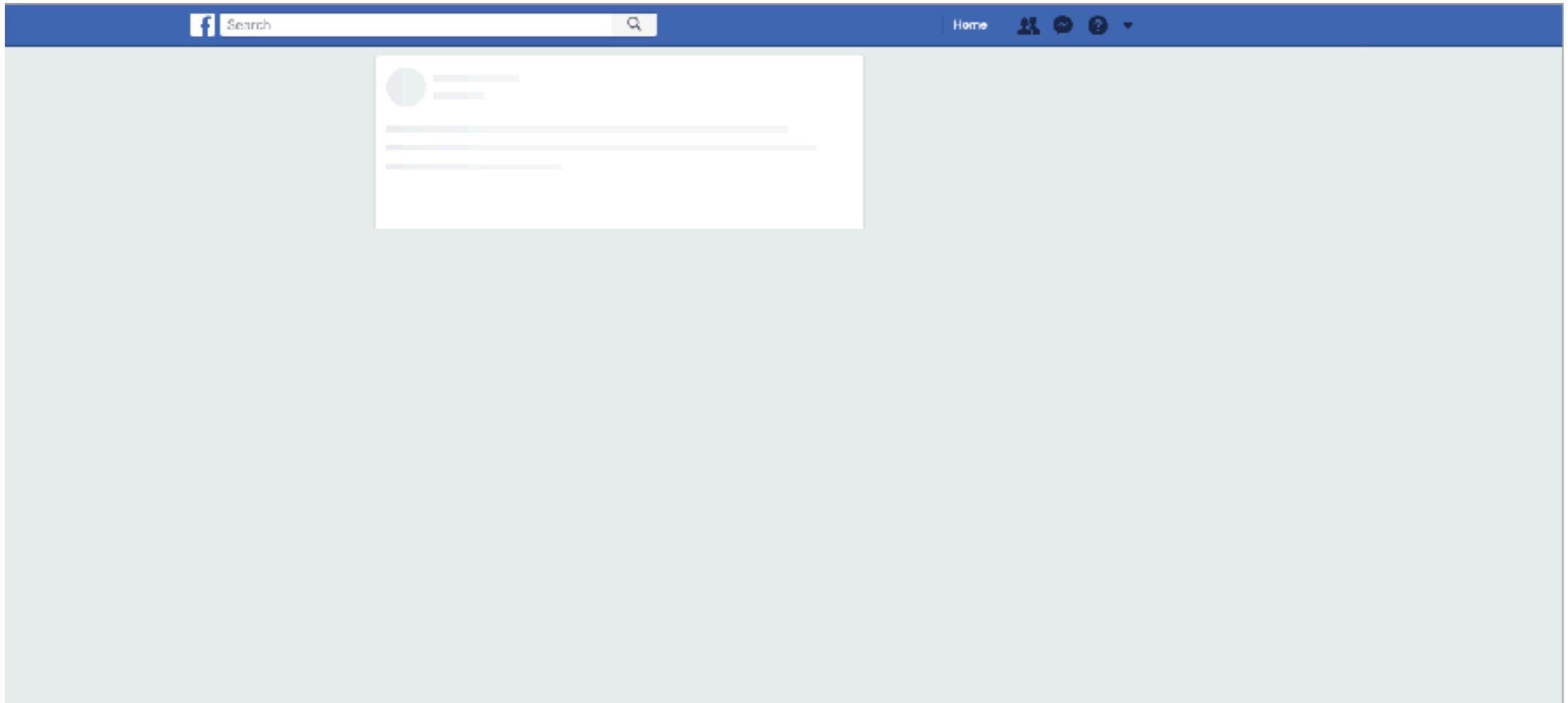


# KEEP YOUR API CREDENTIALS PRIVATE



- Don't post to a public code repo
- Don't share with other developers outside of your organization

# YOUR APP MIGHT EXPERIENCE A DELAYED RESPONSE

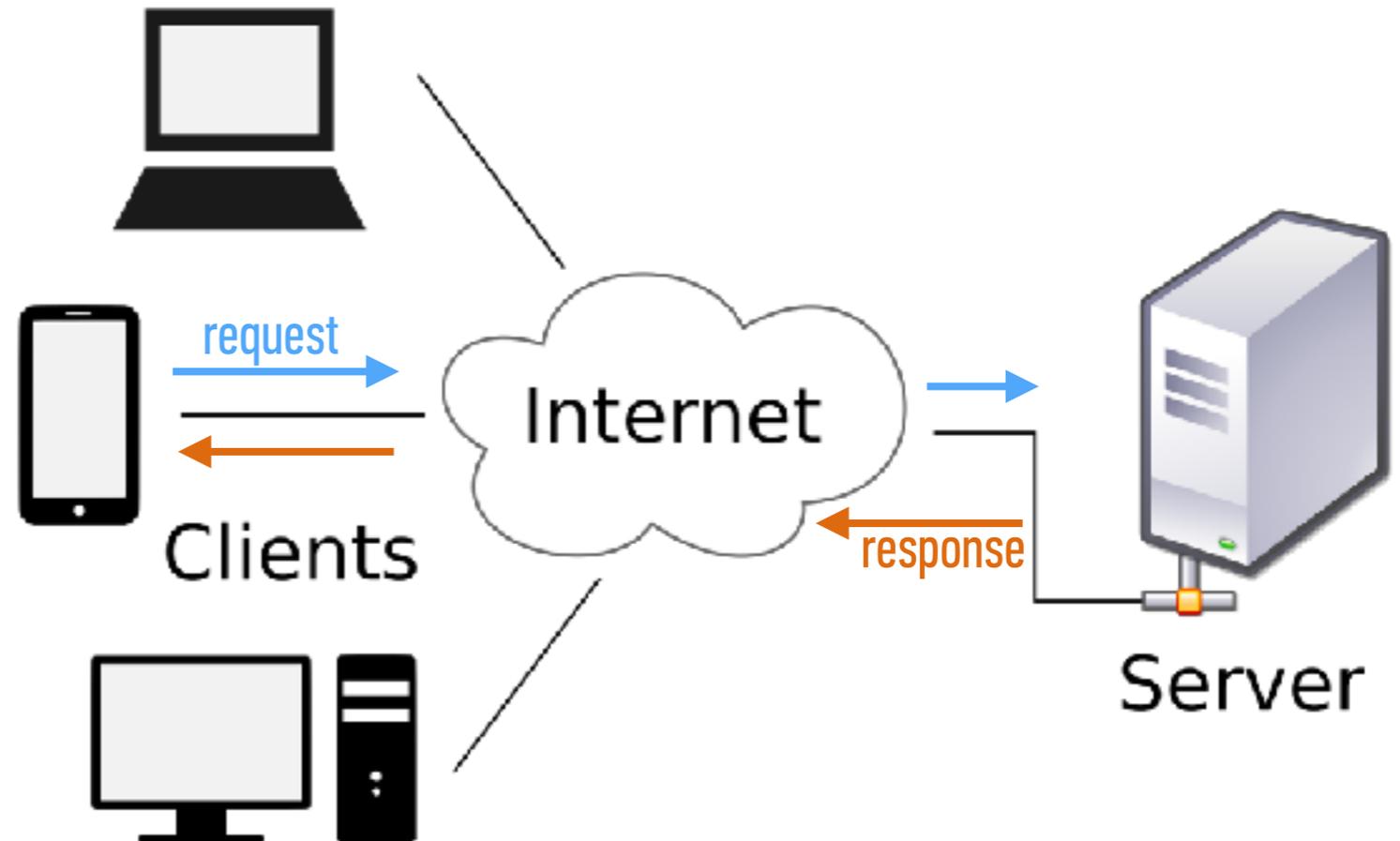


# YOUR REQUEST MAY RESULT IN AN ERROR



# REST (representational state transfer)

- architectural style of web applications
- transfers a representation of the state of a resource between the server and the client



# RESTful API

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

## By geographic coordinates

API call:

`api.openweathermap.org/data/2.5/weather?lat={lat}&lon={lon}`

Parameters:

lat, lon coordinates of the location of your interest

Examples of API calls:

`api.openweathermap.org/data/2.5/weather?lat=35&lon=139`

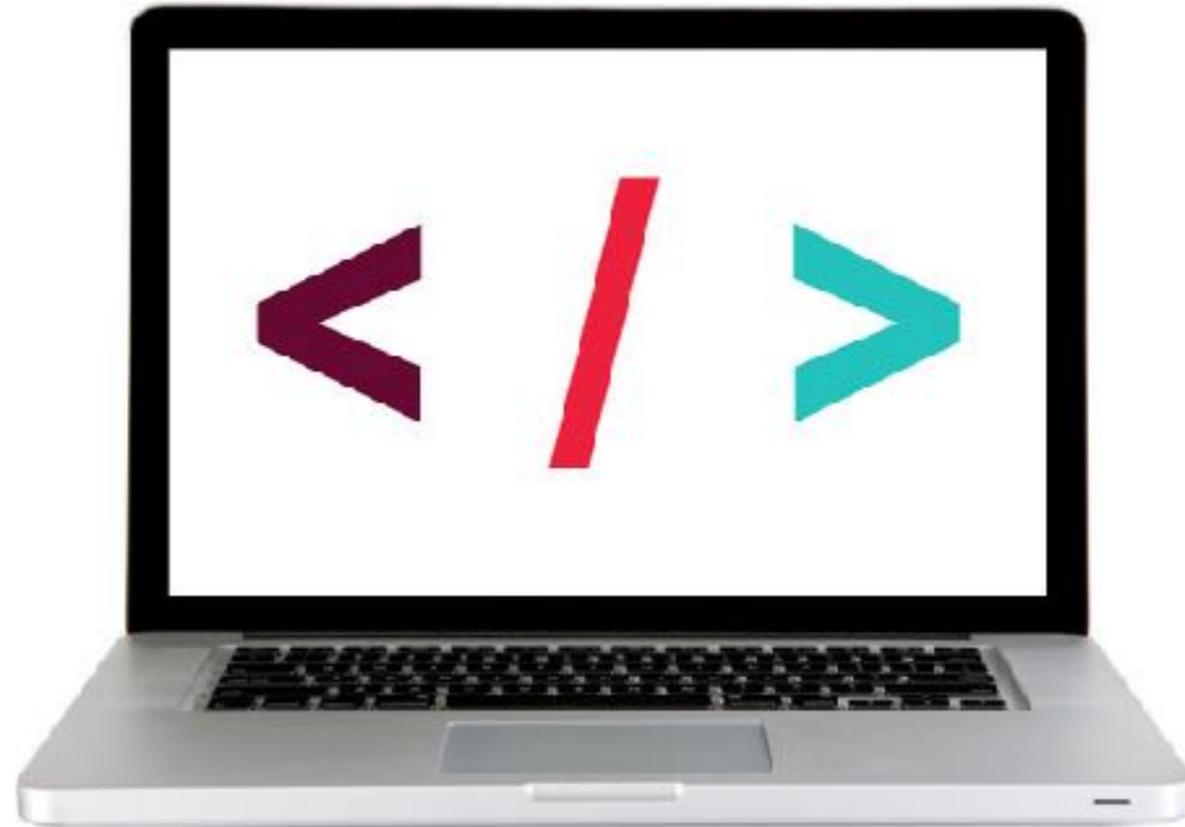
API respond:

```
{
  "coord": {"lon": 139, "lat": 35},
  "sys": {"country": "JP", "sunrise": 1369769524, "sunset": 1369821049},
  "weather": [{"id": 804, "main": "clouds", "description": "overcast clouds", "icon": "04n"}],
  "main": {"temp": 289.5, "humidity": 89, "pressure": 1013, "temp_min": 287.04, "temp_max": 292.04},
  "wind": {"speed": 7.31, "deg": 187.002},
  "rain": {"3h": 0},
  "clouds": {"all": 92},
  "dt": 1369824698,
  "id": 1851632,
  "name": "Shuzenji",
  "cod": 200}
}
```

---

**LET'S TAKE A CLOSER LOOK**

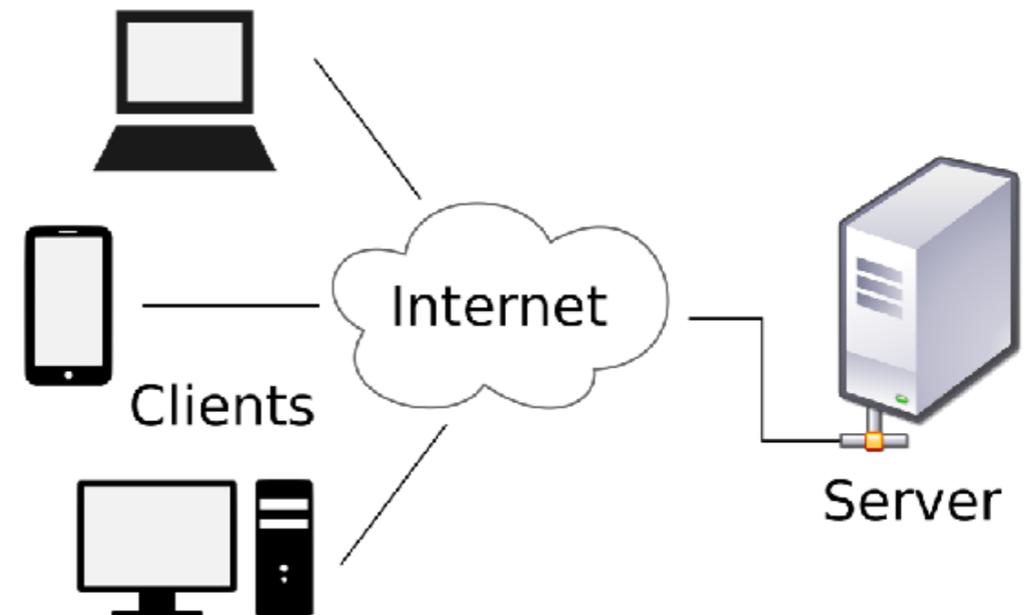
---



# HTTP

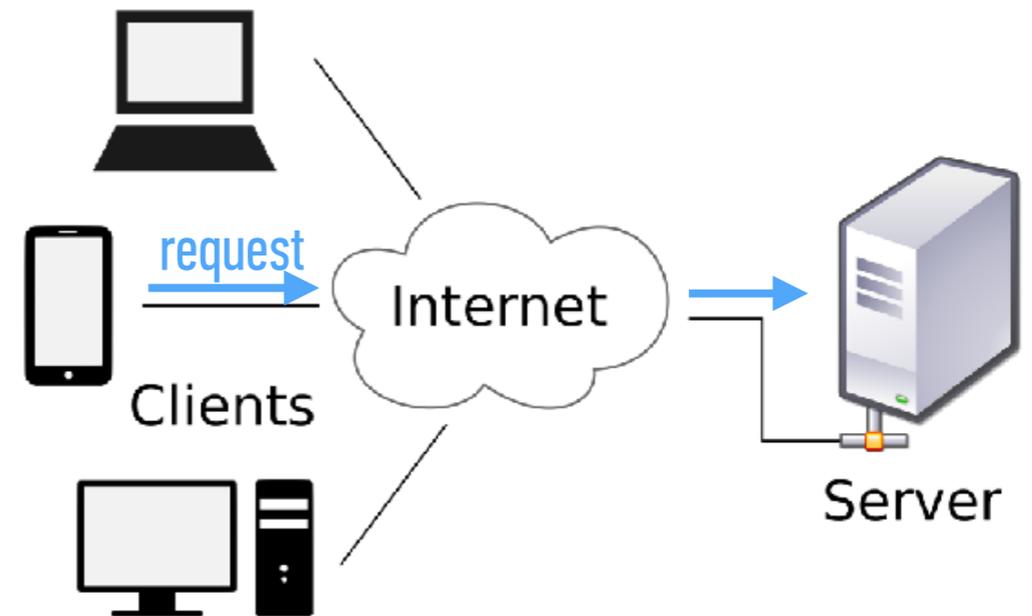
# HTTP (hypertext transfer protocol)

- System of rules for how web pages are transmitted between computers
- Defines the format of messages passed between HTTP clients and HTTP servers



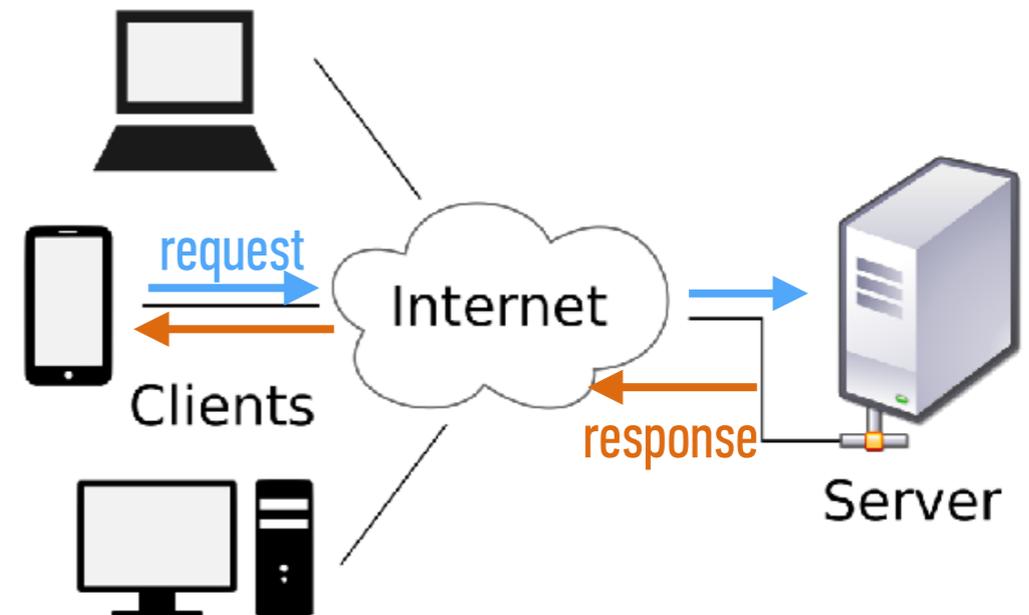
# HTTP (hypertext transfer protocol)

- A client sends a **request** to a server.

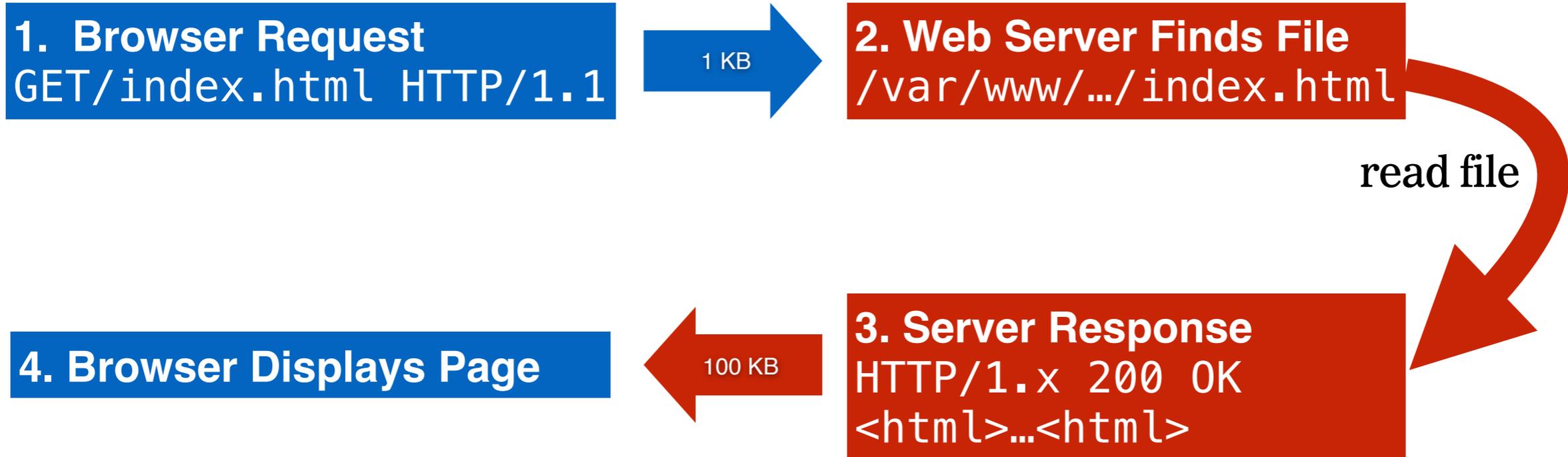


# HTTP (hypertext transfer protocol)

- A server sends a **response** back to a client.



# HTTP REQUEST AND RESPONSE



# HTTP (hypertext transfer protocol)

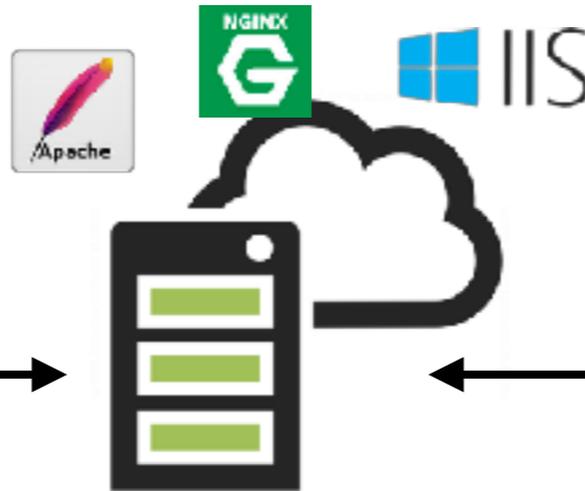
HTTP client

web browser



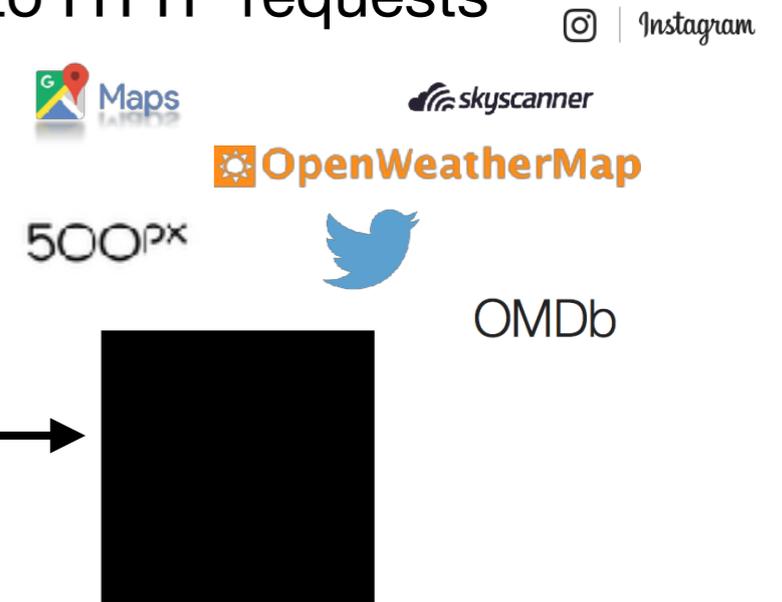
HTTP server

web server software



Web service

app that responds to HTTP requests



# HTTP REQUESTS IN EVERYDAY LIFE

protocol

host

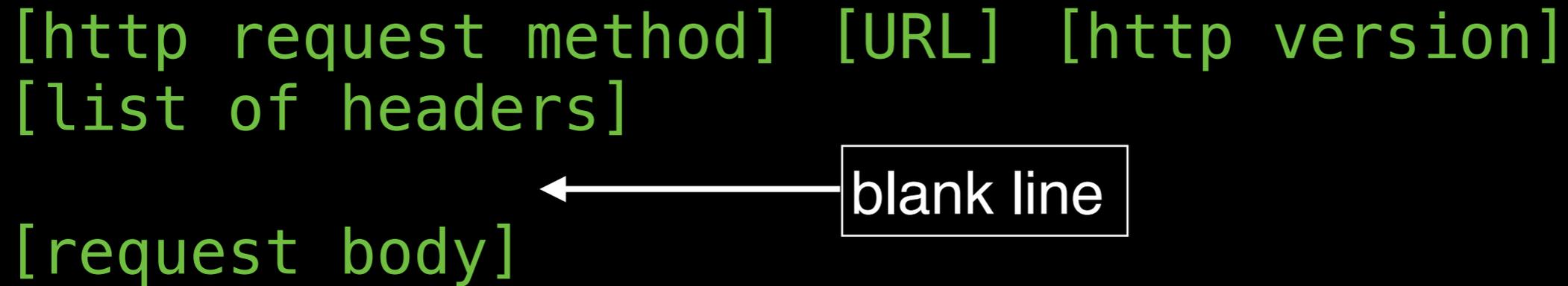
resource path

query

`https://www.domain.com/path/to/resource?a=b&x=y`

# HTTP REQUEST STRUCTURE

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



The diagram shows the structure of an HTTP request. It consists of three lines of text in a monospace font, all in green. The first line is "[http request method] [URL] [http version]". The second line is "[list of headers]". The third line is "[request body]". A white arrow points from a white box labeled "blank line" to the space between the second and third lines. Another white arrow points from a white box labeled "optional" to the "[request body]" line.

blank line

optional

# HTTP REQUEST METHODS (“HTTP VERBS”)

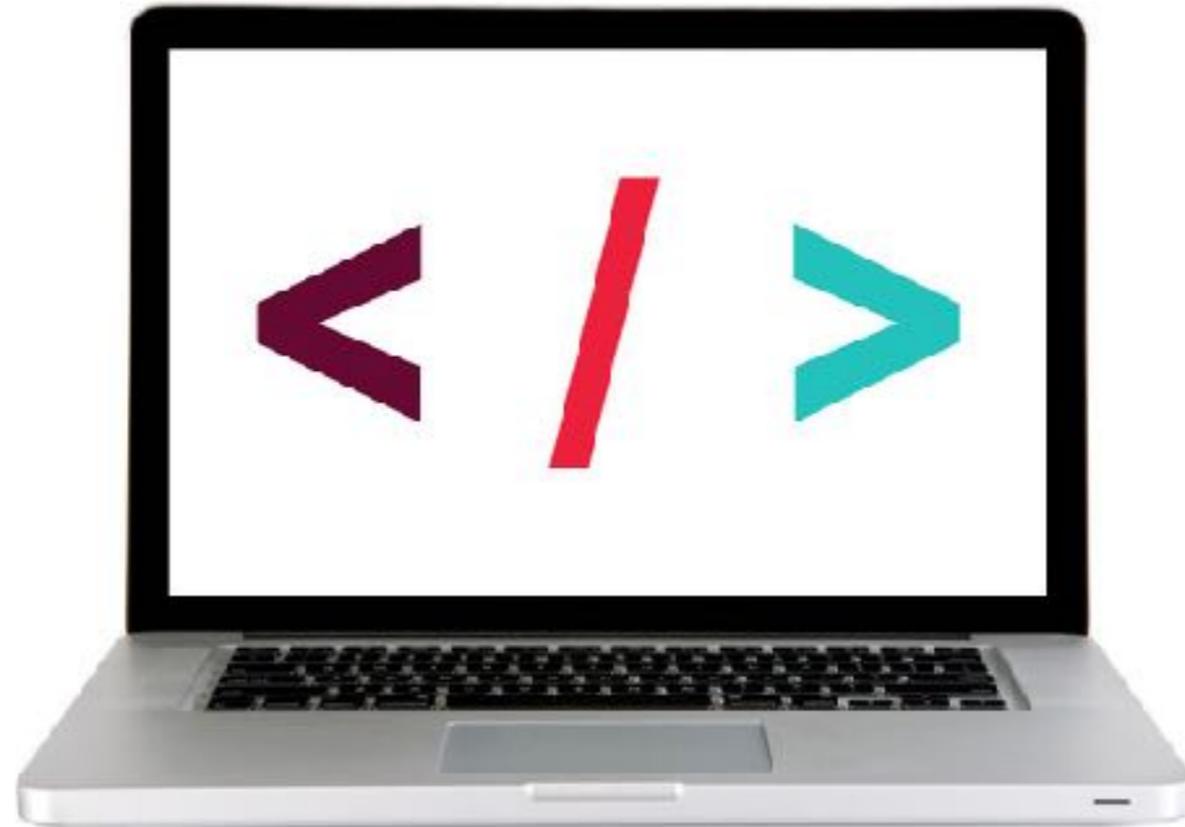
GET	Retrieve a resource
POST	Create a resource
PATCH	Update an existing resource
PUT	Replace an existing resource
DELETE	Delete a resource

Most widely used

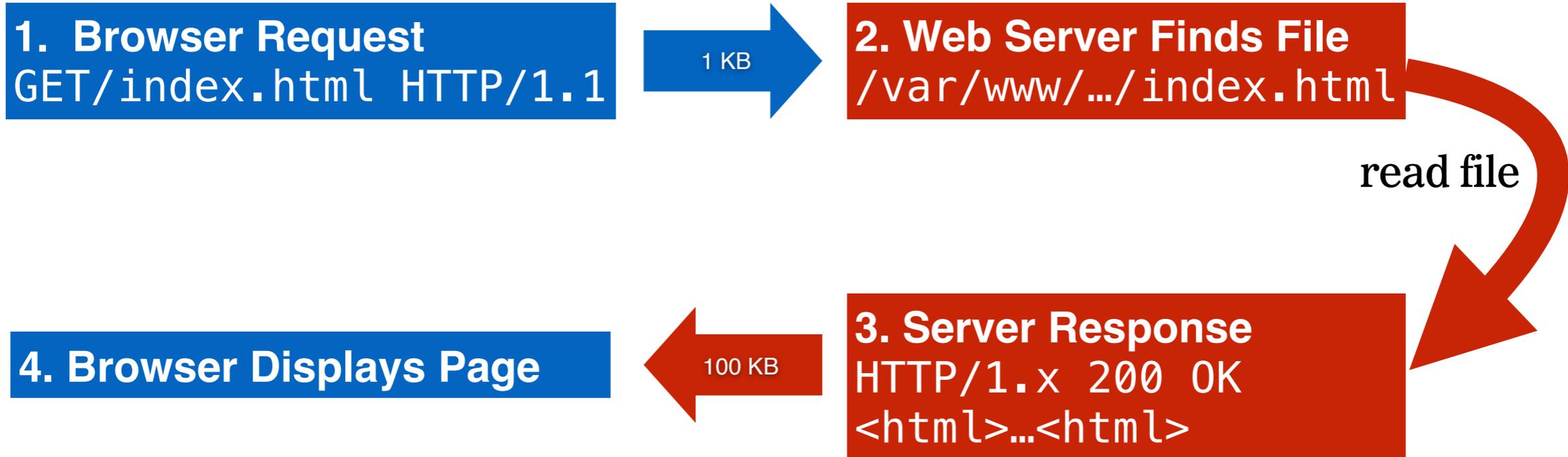
---

**LET'S TAKE A CLOSER LOOK**

---



# HTTP REQUEST AND RESPONSE



# HTTP RESPONSE STRUCTURE

```
[http version] [status] [reason]  
[list of headers]  
[response body]
```

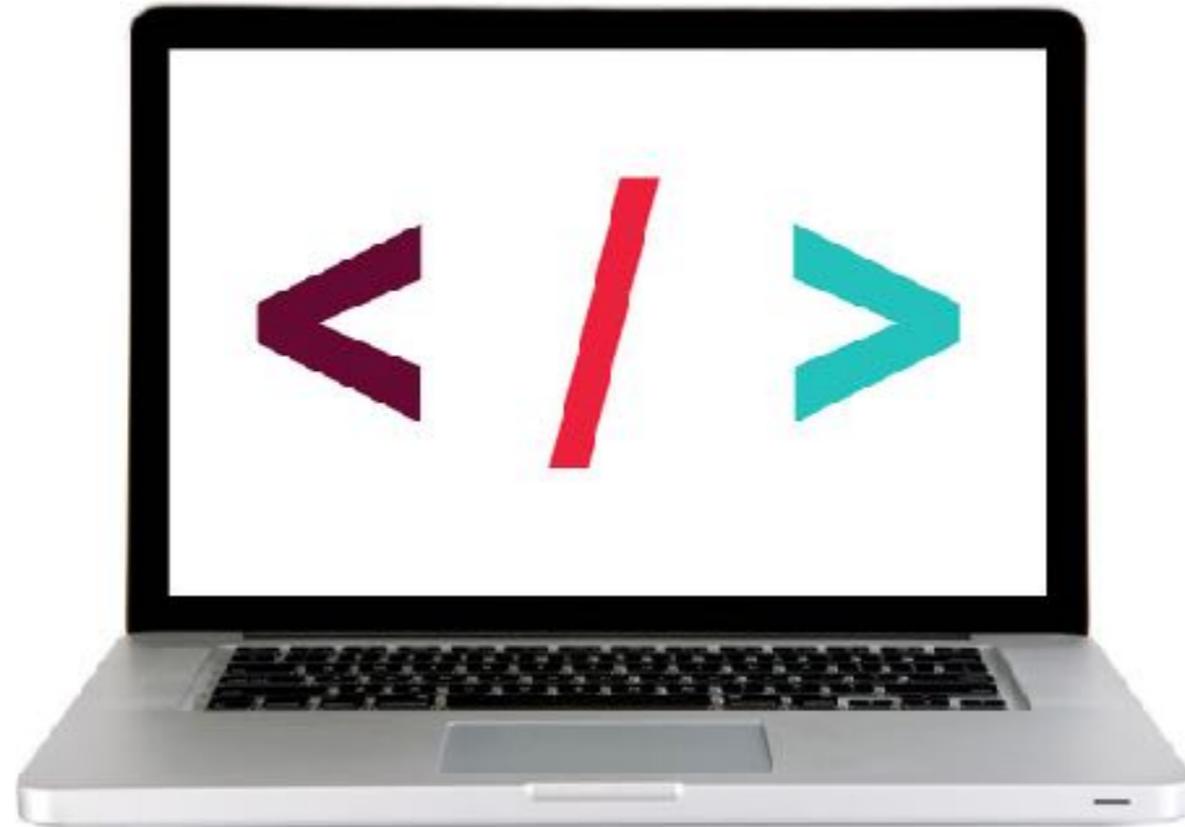
← blank line

← usually HTML, JSON, etc

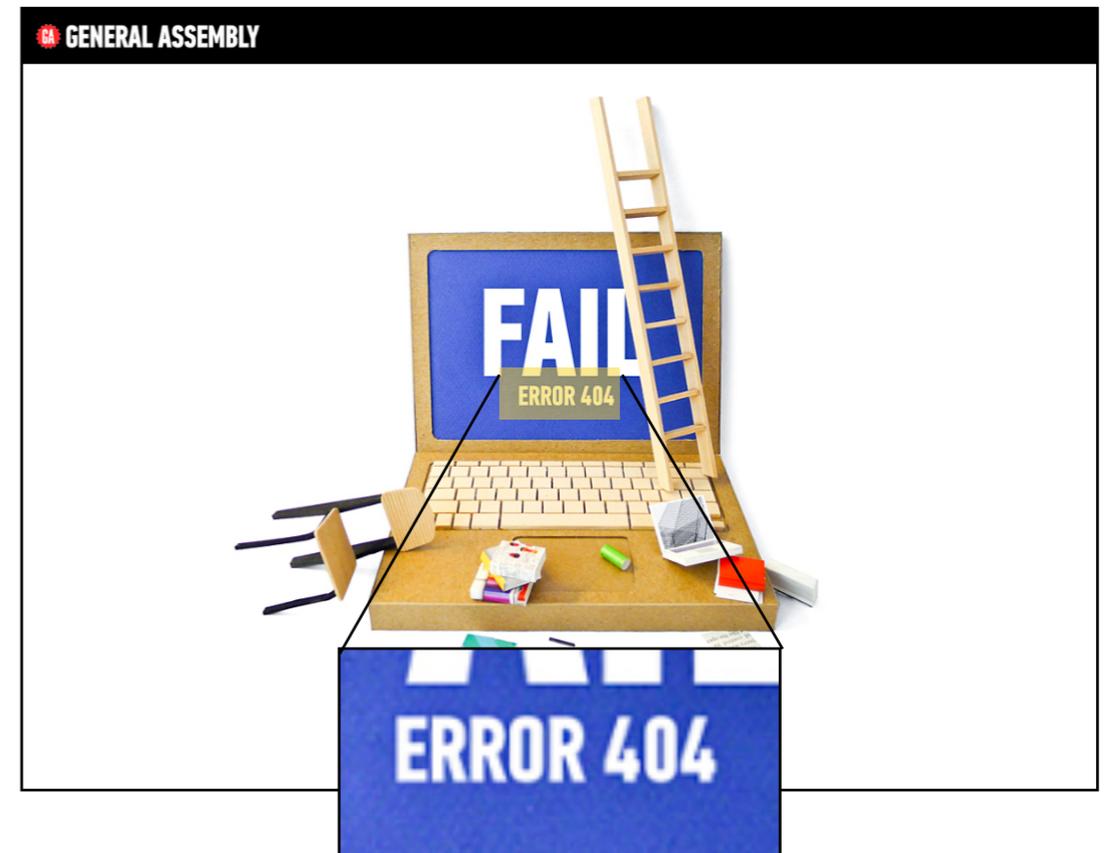
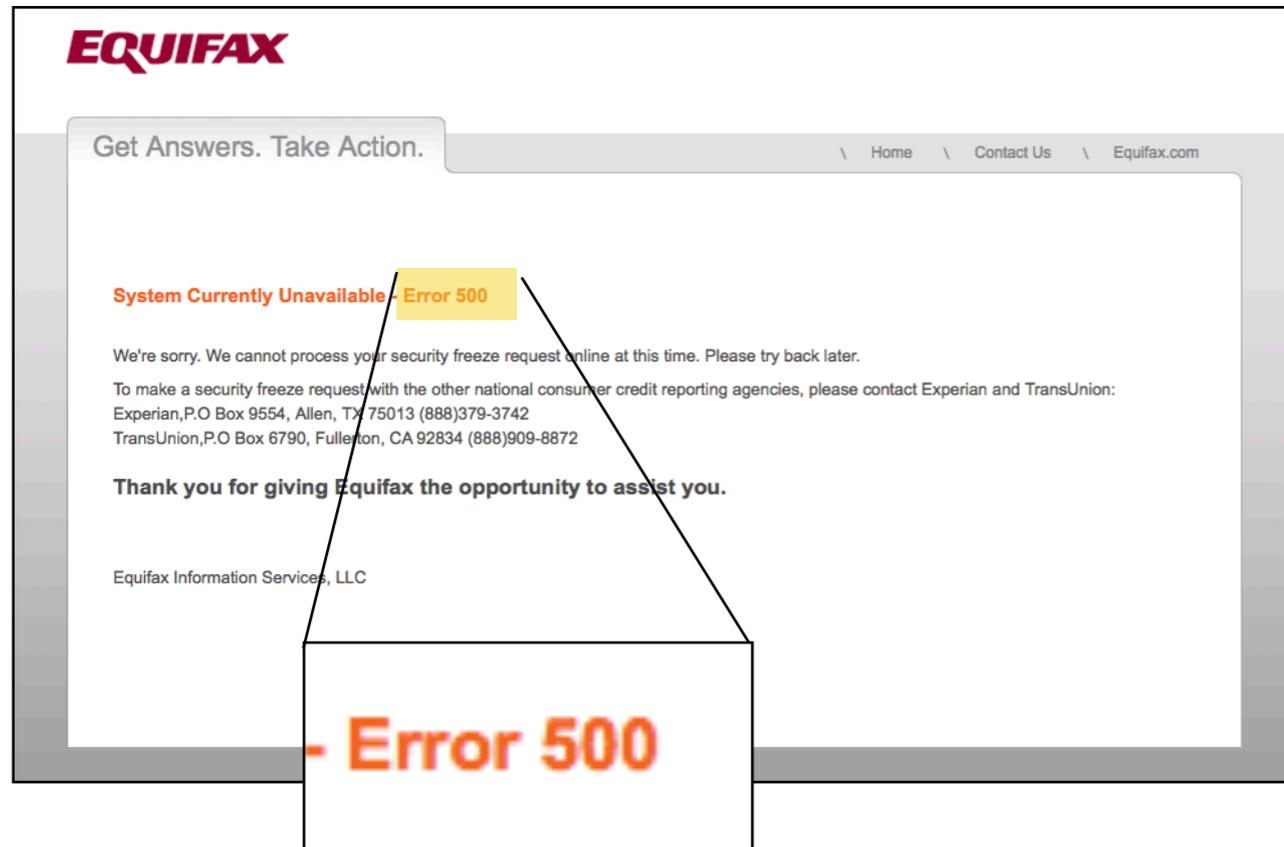
---

**LET'S TAKE A CLOSER LOOK**

---



# HTTP STATUS CODES



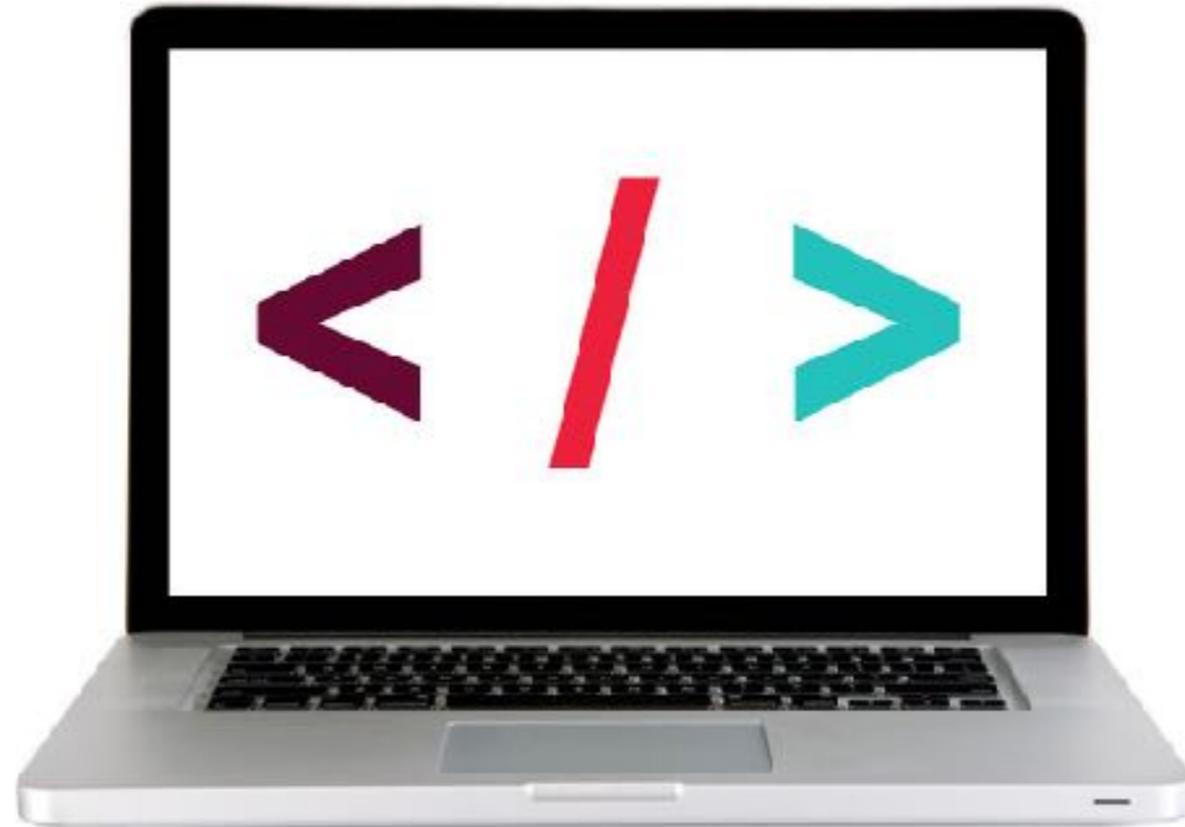
# HTTP STATUS CODES

200	Okay
301	Moved permanently
302	Moved temporarily
400	Bad request
403	Forbidden
404	Not found
418	I'm a teapot
500	Internal server error

---

**LET'S TAKE A CLOSER LOOK**

---



# Ajax

# Ajax

**A** synchronous  
**J**avaScript  
**A**nd  
**X**ML **or JSON!**

# **Ajax in vanilla JS**

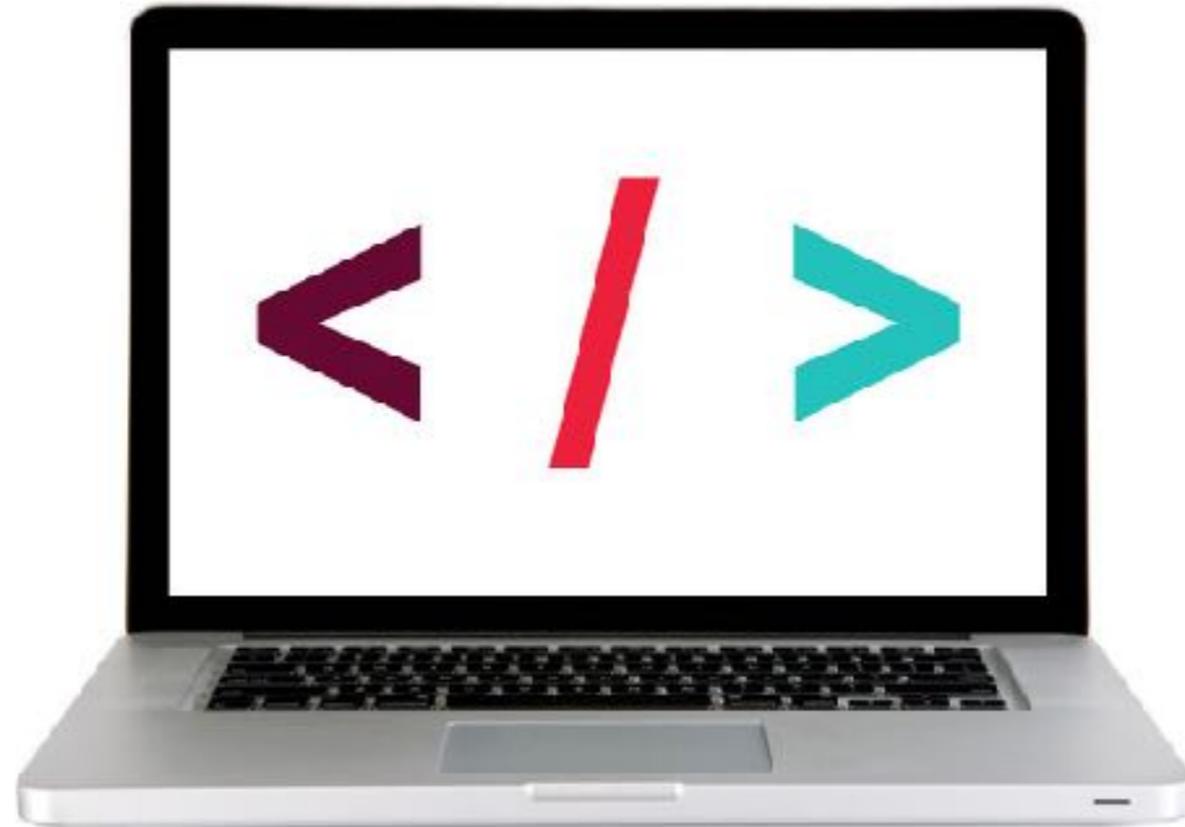
# Fetch = Ajax requests in vanilla JavaScript

```
fetch(url).then(function(response) {  
  // check if request was successful  
}).then(function(data) {  
  // do something with the data  
});
```

---

**LET'S TAKE A CLOSER LOOK**

---



---

# EXERCISE - CREATING AN AJAX REQUEST

---



EXERCISE

## LOCATION

---

▶ starter-code > 1-fetch-ajax-exercise

## TIMING

---

5 *min*

1. Copy the code from the codealong to the main.js file.
2. Change the URL to the one shown in the instructions.
3. Verify that a new set of results is shown in the console.
4. **BONUS:** Customize the error message to display the text of the HTTP status message.  
(Hint: look at <https://developer.mozilla.org/en-US/docs/Web/API/Response/statusText>)
5. **BONUS:** Refactor the code to work with user interaction. In the index.html file there is a "Get Health Data" button. Modify your code so data is only requested and logged to the console after a user clicks the button.

# **jQuery Ajax**

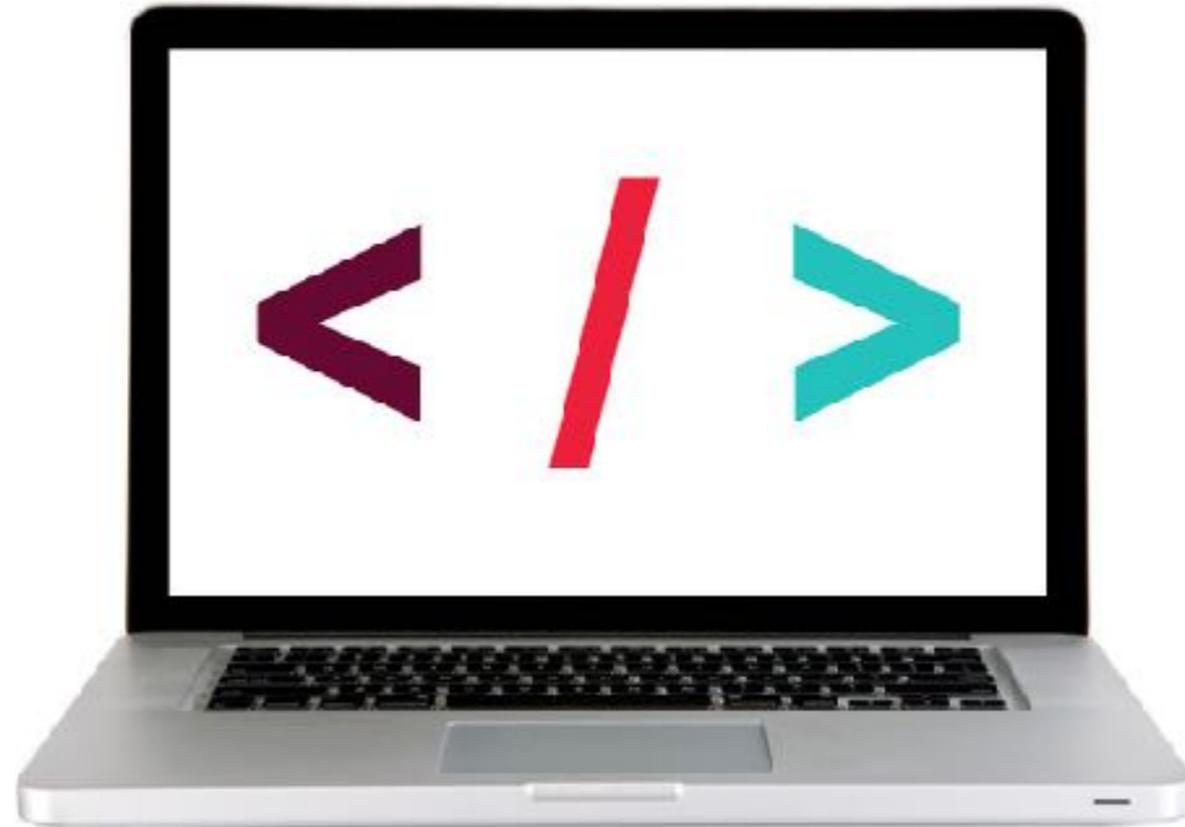
# Using Ajax with jQuery

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

---

**LET'S TAKE A CLOSER LOOK**

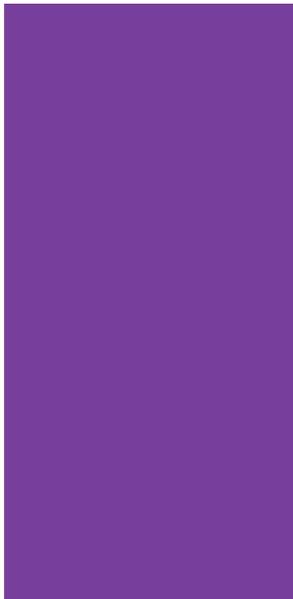
---



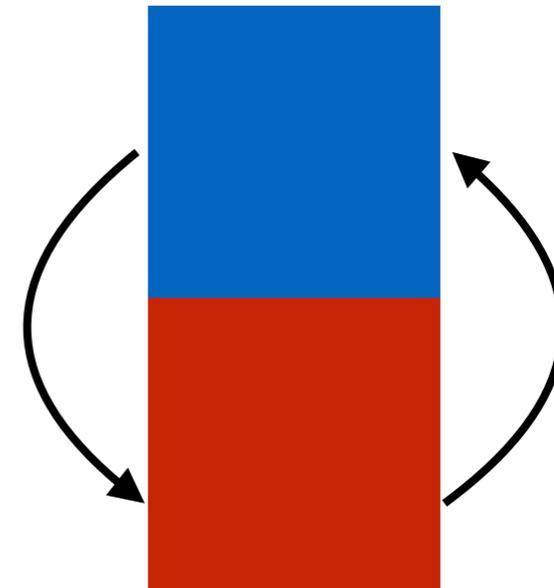
# **Code organization**

# SEPARATION OF CONCERNS

code for data  
and view  
intermingled



code for data



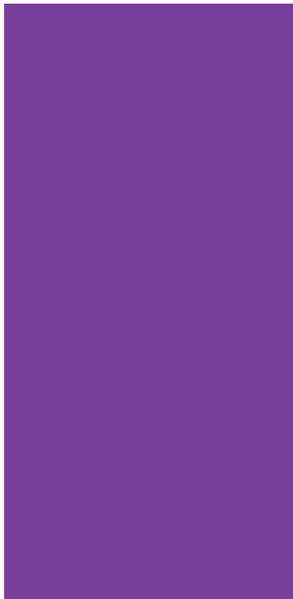
parts of code  
call each  
other, but are  
maintained  
separately

code for view

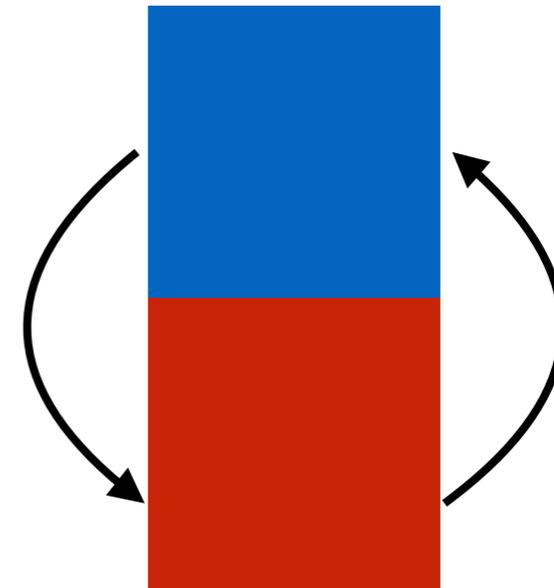


# SEPARATION OF CONCERNS - HTTP

code for client  
and for HTTP  
requests  
intermingled



code for client



parts of code  
call each  
other, but are  
maintained  
separately

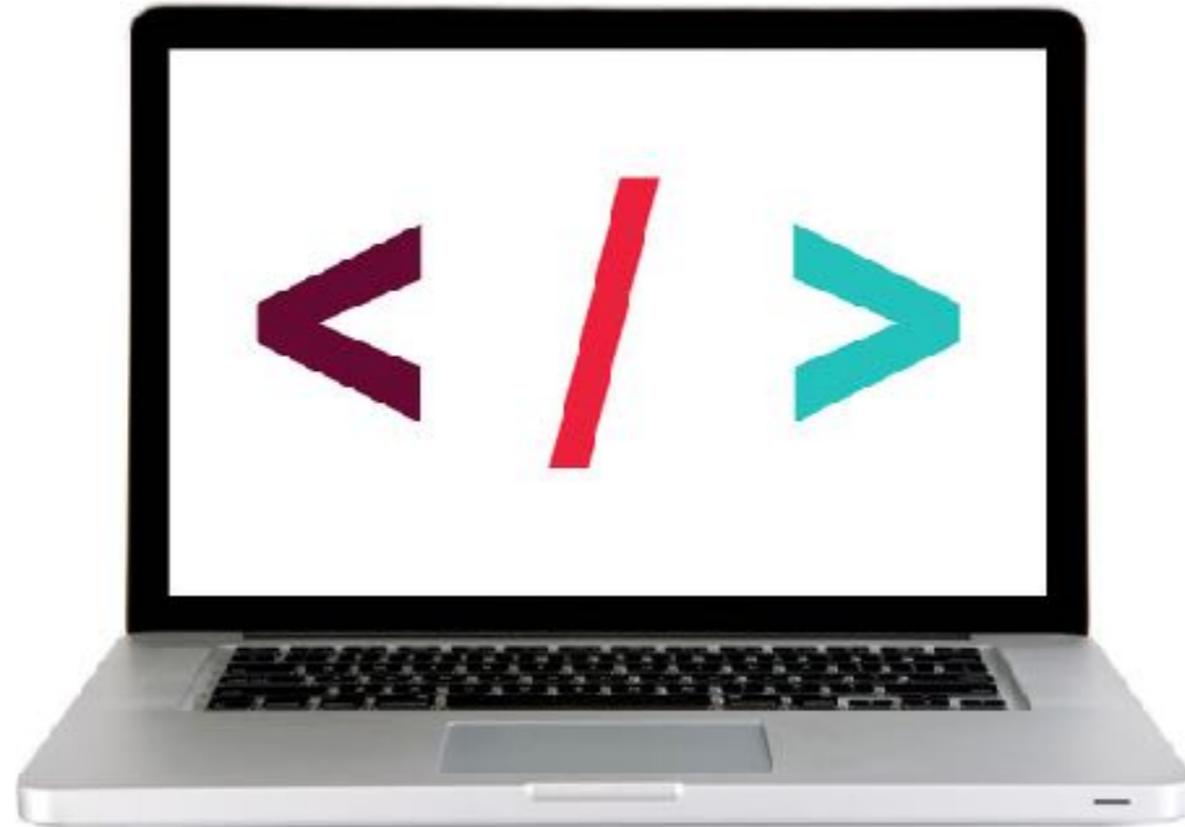
code for HTTP



---

**LET'S TAKE A CLOSER LOOK**

---



# CREATING DRY CODE FOR HTTP REQUESTS

Your app

Web services

Code to get data from source #1 and add to view

Code to get data from source #2 and add to view

Code for HTTP request

Code for HTTP request is separate from code for data parsing and DOM manipulation

Source #1

Source #2

# LAB — JQUERY AJAX

---



## **OBJECTIVE**

---

- ▶ Create an Ajax request using jQuery or Fetch.

## **LOCATION**

---

- ▶ `starter-code > 4-ajax-lab`

## **EXECUTION**

---

*45 min*

1. Open `index.html` in your editor and familiarize yourself with the structure and contents of the file.
2. Open `main.js` in your editor and follow the instructions.

# **Exit Tickets!**

**(Class #9)**

## **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 Fetch.
- Create an Ajax request using jQuery.

## **NEXT CLASS PREVIEW**

### **Asynchronous JavaScript and Callbacks**

- Describe what asynchronous means in relation to JavaScript
- Pass functions as arguments to functions that expect them.
- Write functions that take other functions as arguments.
- Build asynchronous program flow using Fetch

# **Q&A**