

JAVASCRIPT DEVELOPMENT

Sasha Vodnik, Instructor

HELLO!

- 1. Pull changes from the svodnik/JS-SF-16-resources repoto your computer
- 2. Open the 09-ajax-apis/starter-code folder in your code editor

AJAX & ASYNCHRONOUS JAVASCRIPT

LEARNING OBJECTIVES

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

- Use event delegation to manage dynamic content.
- Use implicit iteration to update elements of a jQuery selection
- 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 5

Events & jQuery / Ajax & APIs

WEEK 6

Asynchronous JS & callbacks / Advanced APIs

WEEK 7

Project 2 lab / Prototypal inheritance

EXIT TICKET QUESTIONS

EXIT TICKET QUESTIONS

- 1. What is the best practice for knowing when to use libraries like jQuery. Is it a time and money type of decision and what's the current trend.
- 2. Is there any tips other than knowing syntax, when jquery (i.e. \$ sign), JS, or Vanilla JS is being used?
- 3. There are a lot of new terms and syntax elements that have been introduced over the last few classes. Is there anything we can do to better familiarize with these, aside from coding and general practice?
- 4. You mentioned that less and less projects are using jQuery. What's the library or the practice that is replacing it?
- 5. From an optimization perspective is vanilla javascript always faster without jQuery when loading a webpage.

JQUERY BEST PRACTICES

METHOD CHAINING

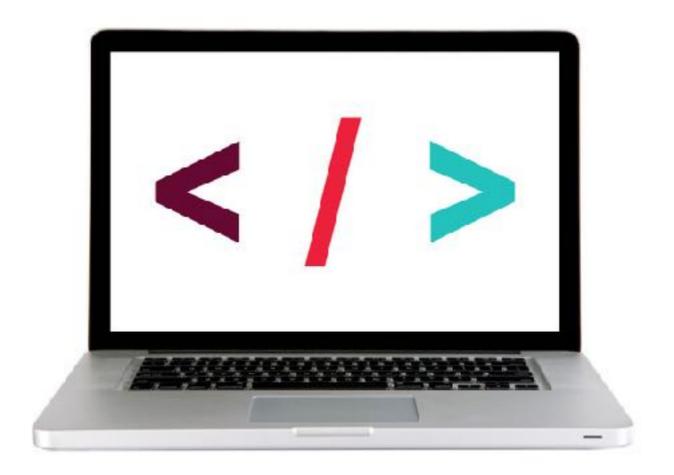
CHAINING

without chaining:

```
let $mainCaption = $('');
let $captionWithText = $mainCaption.text('Today');
let $fullCaption = $captionWithText.addClass('accent');
```

with chaining:

```
let $fullCaption = $('').text('Today').addClass('accent');
```



LET'S TAKE A CLOSER LOOK

IMPLICIT ITERATION

IMPLICIT ITERATION

explicit iteration

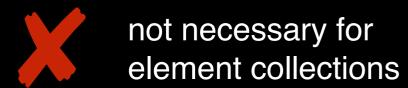
```
$('li').each(function() {
  $(this).removeClass('current');
});
```

jQuery .each() method works like a forEach loop

implicit iteration

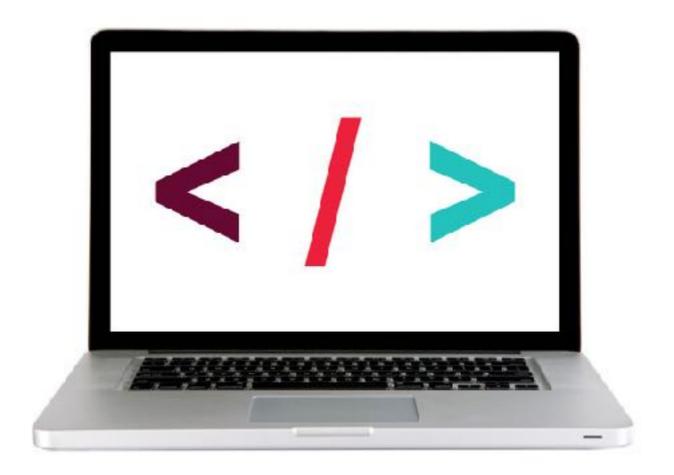
```
$('li').removeClass('current');
```

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





less code = best practice!



LET'S TAKE A CLOSER LOOK

EXERCISE - IMPLICIT ITERATION



OBJECTIVES

- Use chaining to place methods on selectors.
- Use implicit iteration to update elements of a jQuery selection.

LOCATION

starter-code > 09-best-practices-exercise

TIMING

5 min

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

EVENT DELEGATION

WITHOUT EVENT DELEGATION

1. load page

2. set event listener on list items

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

- item1item2
- •item3

item1item2item3

```
click event
click event
click event
```

3. add a new list item

item1item2item3item4

click event click event click event

click event is not automatically applied to the new li element



WITH EVENT DELEGATION

1. load page

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

3. add a new list item

```
•item1
•item2
•item3
```

```
selector
changed to
parent

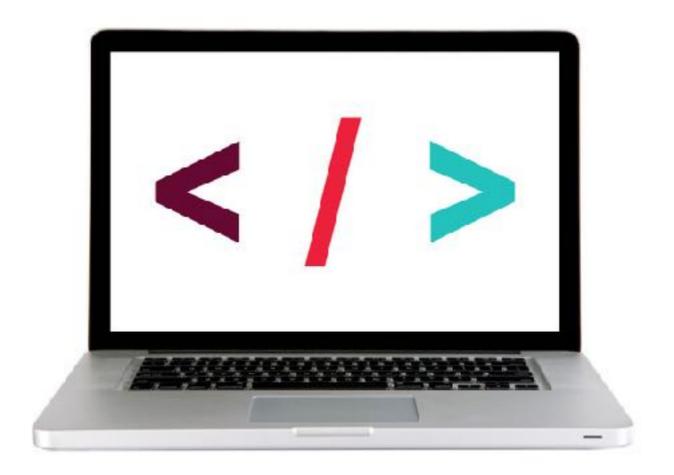
$('ul').on('click', 'li', function(){
addClass('selected')
});

•item1
•item2
•item3

click event
click event
click event
```

```
item1item2item2item3item4
```

click event IS automatically applied to the new 1i element!



LET'S TAKE A CLOSER LOOK

EXERCISE - EVENT DELEGATION



OBJECTIVE

▶ Use event delegation to manage dynamic content.

LOCATION

▶ starter-code > 09-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.

AJAX & APIS

ACTIVITY



TYPE OF EXERCISE

Individual/Partner

TIMING

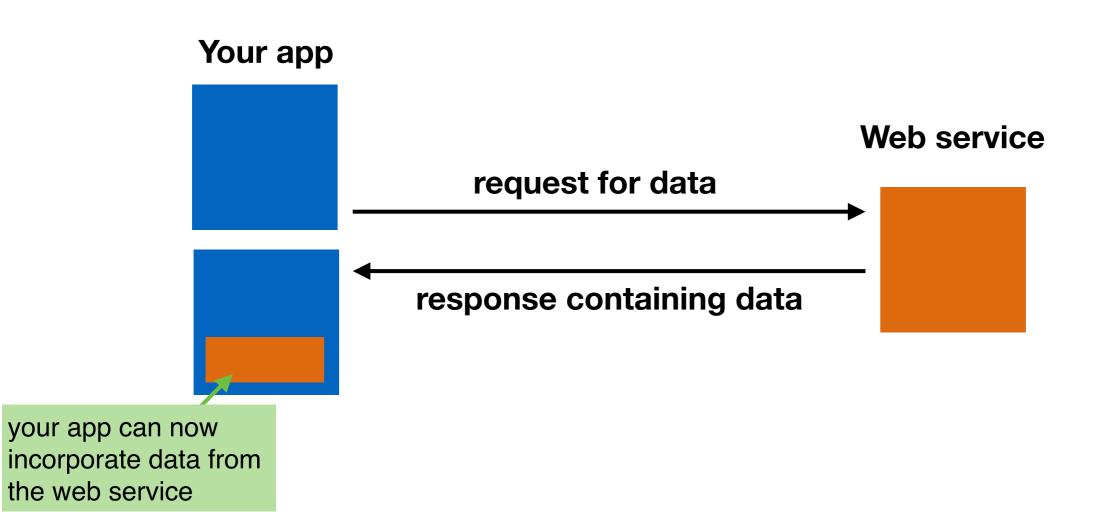
3 min

- 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

AJAX & APIS

WEB SERVICES



my website content

Content from Twitter added using Twitter API



Instructor and Author on Programming and Technology

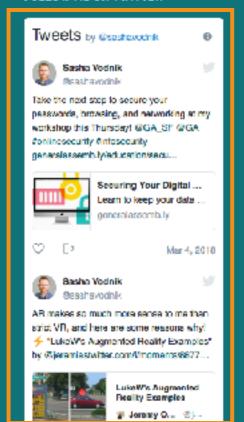
Home

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 --
Sharettls:

FOLLOW ME ON TWITTER



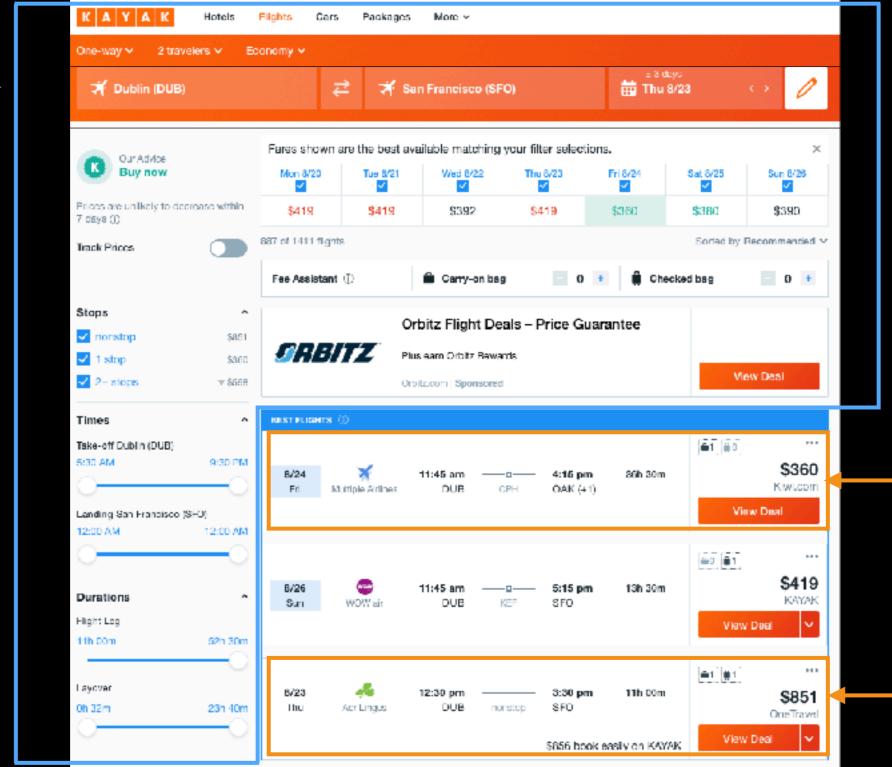
Securing Your Digital Life, Part 1: Choosing a password manager

JANUARY 11, 2018

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



Kayak website content



Content from kiwi.com using API

Content from OneTravel using API

AJAX & APIS

WEB SERVICES





OpenWeatherMap







Instagram



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

By geographic coordinates

API call:

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)
 - JavaScript libraries (jQuery API)
 - services (Slack)











AJAX & APIS

WEB SERVICES





OpenWeatherMap







Instagram



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/

ENDPOINTS

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, Ion 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":"0
    4n"}],
    "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},
    "d+":1360824608
```

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

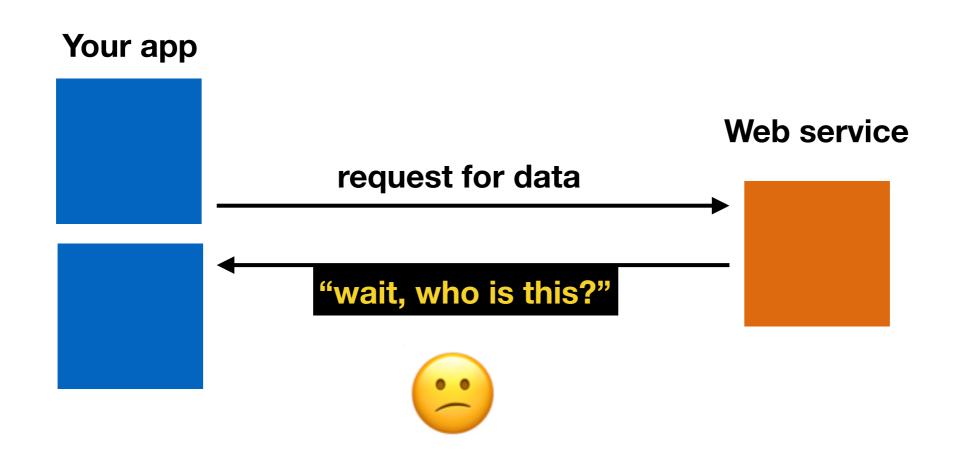
AJAX & APIS

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

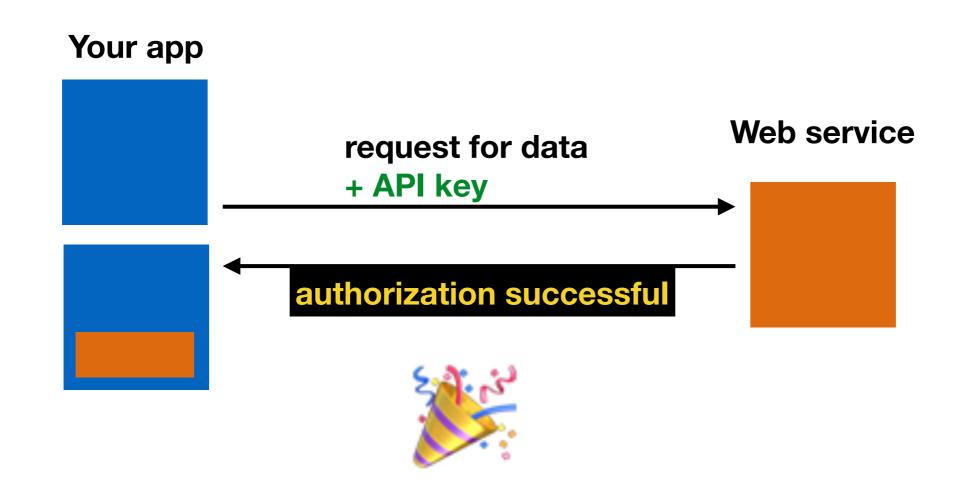


API KEY



- unguessable character string
- connects your requests to your account

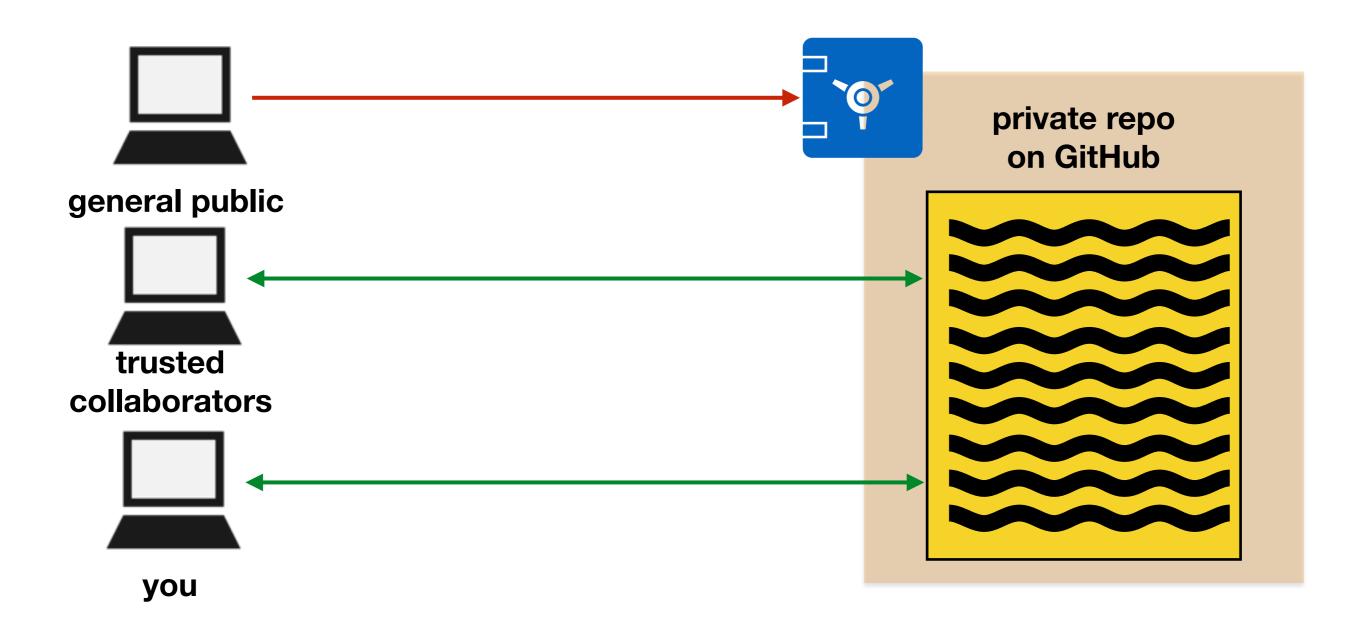
API REQUEST WITH AUTHENTICATION

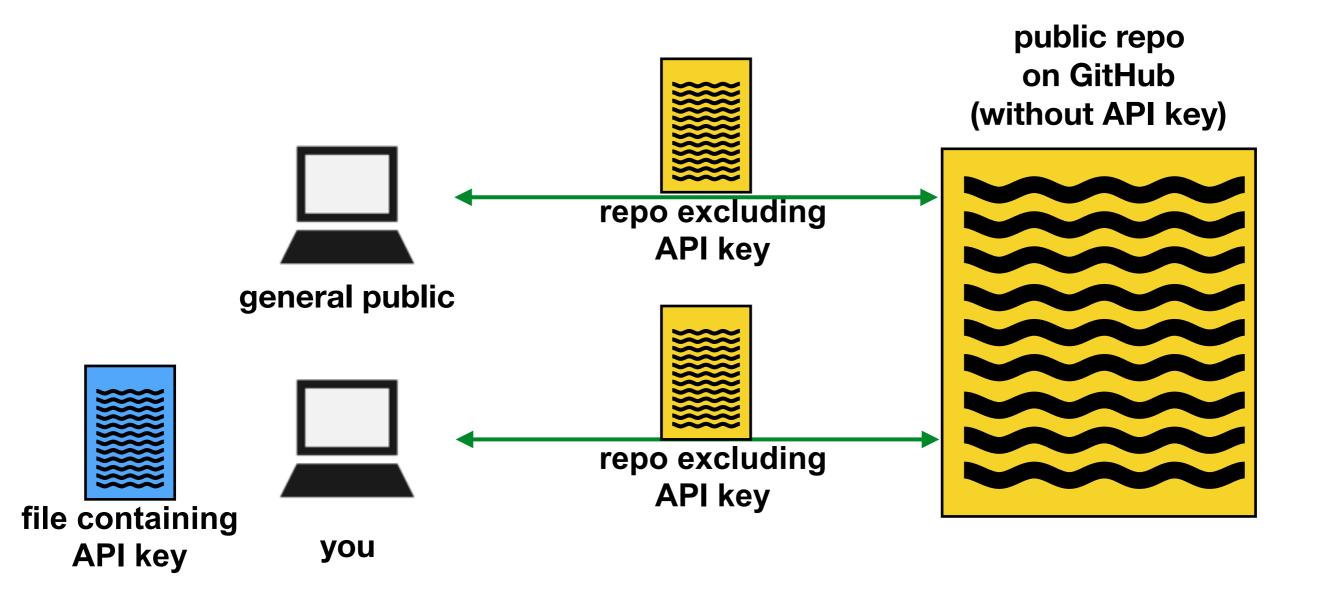


KEEP YOUR API CREDENTIALS PRIVATE



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

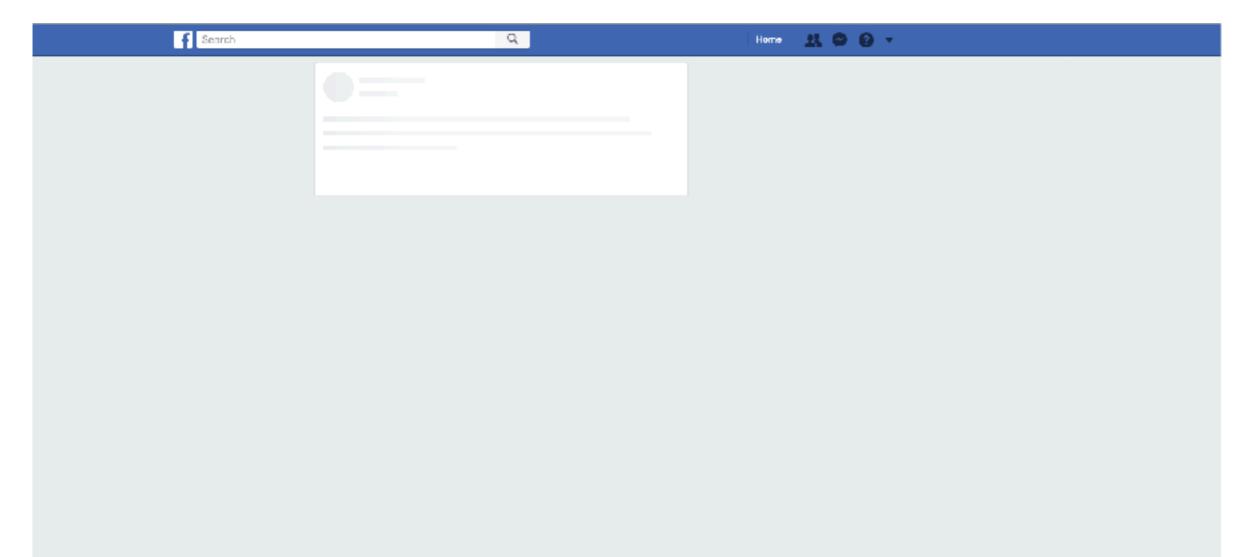




.GITIGNORE

```
js/keys.js
```

YOUR APP MIGHT EXPERIENCE A DELAYED RESPONSE

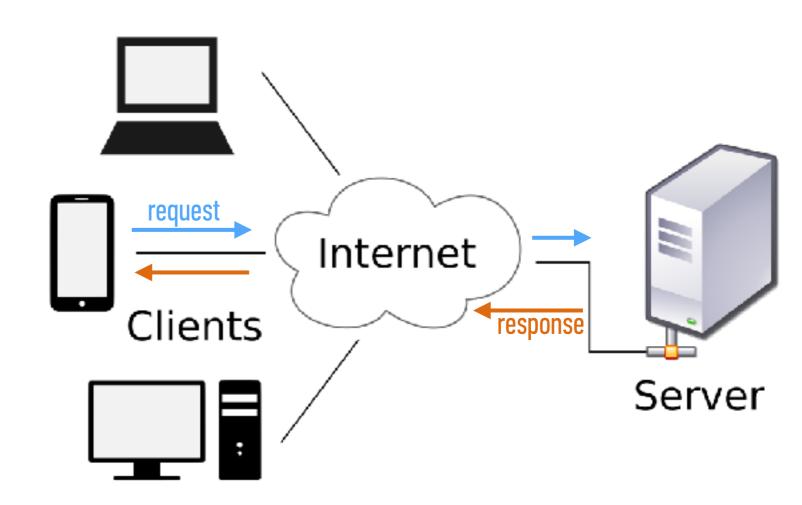


YOUR REQUEST MAY RESULT IN AN ERROR



REST (representational state transfer)

- architectural style of web applications
- 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, Ion coordinates of the location of your interest

Examples of API calls:

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

API respond:

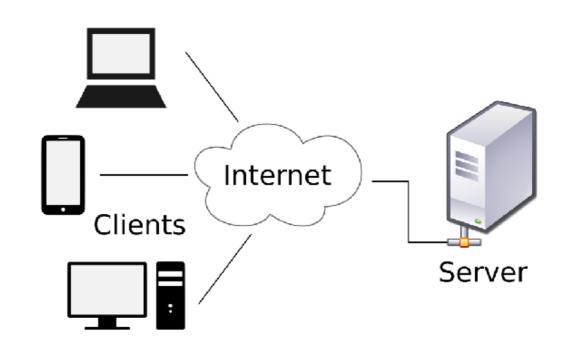
```
{"coord":{"lon":139,"lat":35},
"sys":{"country":"JF","sunrise":1369769524,"sunset":1369821049},
"weather":[{"id":804,"main":"clouds","description":"overcast clouds","icon":"0
4n"}],
"main":{"temp":289.5,"humidity":89,"pressure":1013,"temp_min":287.04,"temp_max
":292.04},
"wind":{"speed":7.31,"deg":187.002},
"rain":{"3h":0},
"clouds":{"all":92},
"dt":1369824698,
"id":1851632,
"name":"Shuzenji",
"cod":200}
```

LET'S TAKE A CLOSER LOOK



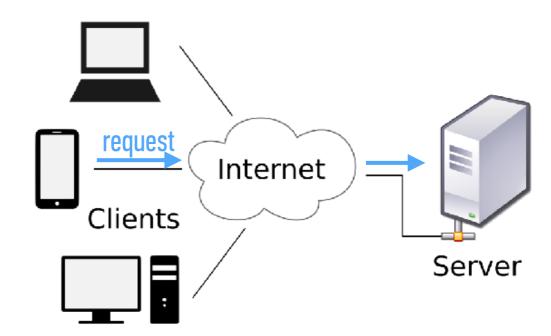
HTTP (hypertext transfer protocol)

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



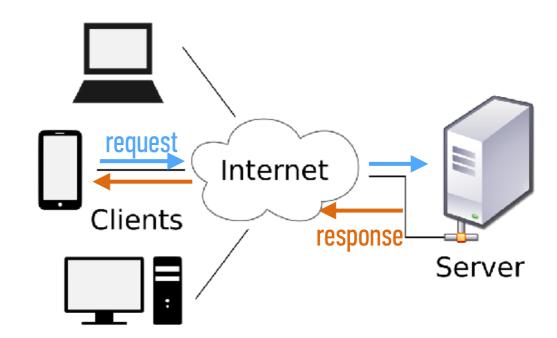
HTTP (hypertext transfer protocol)

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



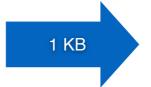
HTTP (hypertext transfer protocol)

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



HTTP REQUEST AND RESPONSE

1. Browser Request
GET/index.html HTTP/1.1



2. Web Server Finds File
/var/www/.../index.html

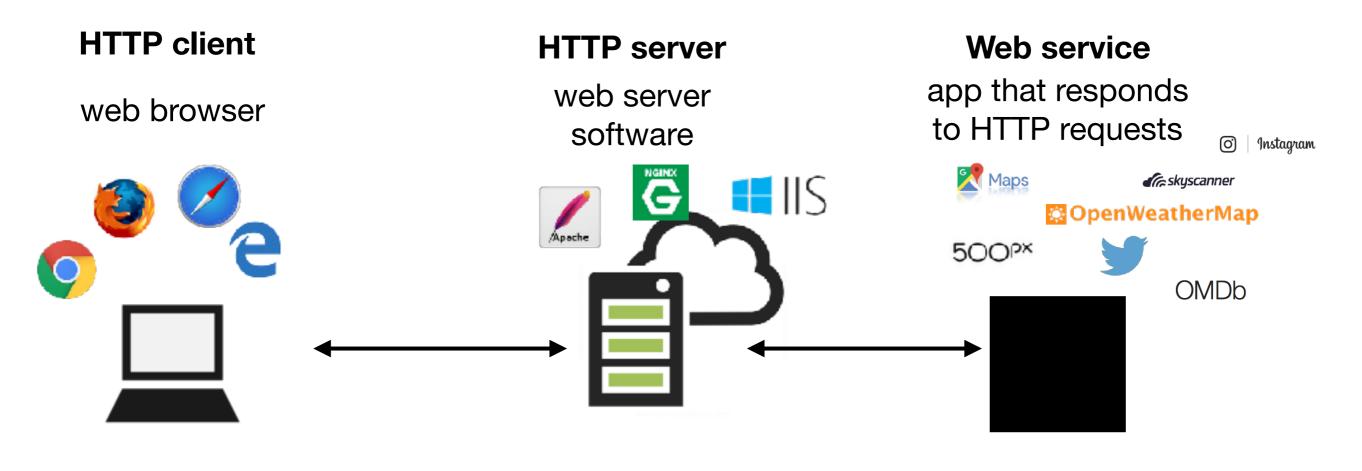
read file

4. Browser Displays Page



3. Server Response HTTP/1.x 200 OK https://www.ntml

HTTP (hypertext transfer protocol)



HTTP REQUESTS IN EVERYDAY LIFE





resource path



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

HTTP REQUEST STRUCTURE

HTTP REQUEST METHODS ("HTTP VERBS")

GET	Retrieve a resource
P0ST	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

1. Browser Request
GET/index.html HTTP/1.1



2. Web Server Finds File
/var/www/.../index.html

read file

4. Browser Displays Page



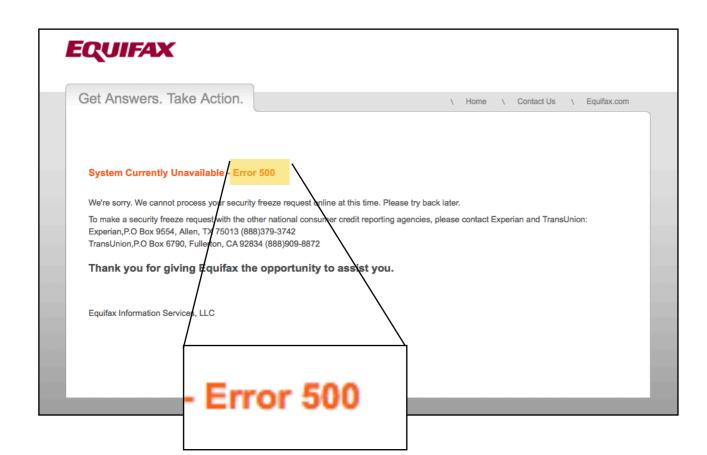
3. Server Response HTTP/1.x 200 OK https://www.ntml

HTTP RESPONSE STRUCTURE

LET'S TAKE A CLOSER LOOK



HTTP STATUS CODES





HTTP STATUS CODES

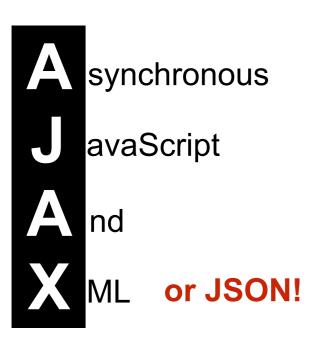
200	Okay
301	Moved permanently
302	Moved temporarily
400	Bad request
403	Forbidden
404	Not found
418	I'm a teapot
500	Internal server error

LET'S TAKE A CLOSER LOOK



Ajax

Ajax



Ajax in vanila JS

Fetch = Ajax requests in vanilla JavaScript

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

LET'S TAKE A CLOSER LOOK



EXERCISE - CREATING AN AJAX REQUEST



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.

Query Ajax

Using Ajax with jQuery

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

LET'S TAKE A CLOSER LOOK



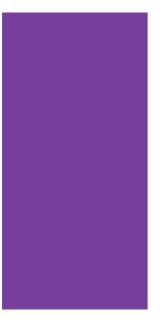
Code organization

LET'S TAKE A CLOSER LOOK



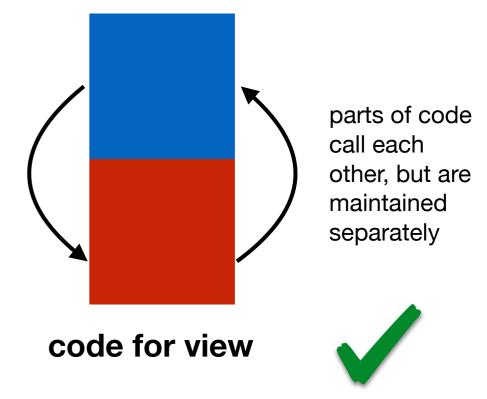
SEPARATION OF CONCERNS

code for data and view intermingled



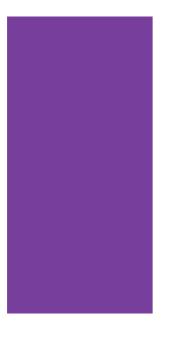


code for data



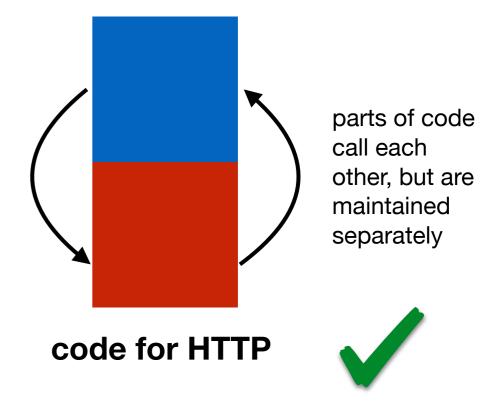
SEPARATION OF CONCERNS - HTTP

code for client and for HTTP requests intermingled





code for client



LAB — JQUERY AJAX



OBJECTIVE

• Create an Ajax request using jQuery or Fetch.

LOCATION

starter-code > 4-ajax-lab

EXECUTION

until 9:20

- 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

- Use event delegation to manage dynamic content.
- Use implicit iteration to update elements of a jQuery selection
- 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
- Integrate string and variable values using template literals

QSA