GENERAL ASSEMBLY

JAVASCRIPT DEVELOPMENT

Sasha Vodnik, Instructor

HELLO!

1. Pull changes from the `svodnik/JS-SF-8-resources` repo to your computer
2. Open the `starter-code` folder in your code editor

AJAX & APIS

LEARNING OBJECTIVES

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

- Use event delegation to manage dynamic content in jQuery.
- 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

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

WEEKLY OVERVIEW

WEEK 6	Advanced jQuery / Ajax & APIs
WEEK 7	Asynchronous JavaScript & Callbacks / Advanced APIs
WEEK 8	Project 2 Lab / Context and `this`

EXIT TICKET QUESTIONS

1. Why `$('ul').on('click', 'li', function(){})` and not `$('ul li').on('click', function(){})`?
2. What kinds of arguments can be passed into the .on() method
3. -I think I got a bit lost on refactoring vanilla JS to jQuery...is this a useful skill to have? How often would I be refactoring in the real world?
   -Is there any resources for refactoring code? It seems to be a popular interview question.

EXIT TICKET QUESTIONS

4. I still don't know if I understand why in the to-do list exercise, to get the strikethrough effect we needed to nest an event inside another event. And why would it not have worked if the events were not nested.
5. Is it best practices to always have every selector as its own variable? I'm used to have a lot of selectors directly select the CSS element it applies to. It's confusing for me now. The hierarchy of references I think is tough CSS->selector->variable->variable going into another variable

INTERMEDIATE VARIABLES VS CHAINING

```
let $newLink = $('<span>');
let $linkWithText = $newLink.html(' complete task');
let $completedLink = $linkWithText.addClass('complete-task');
$(this).append($completedLink);
```

```
let $completedLink = $(‘<span>’).html(‘ complete task’).addClass(‘complete-task’);
$(this).append($completedLink);
```

```
$(this).append($(‘<span>’).html(‘ complete task’).addClass(‘complete-task’));
```

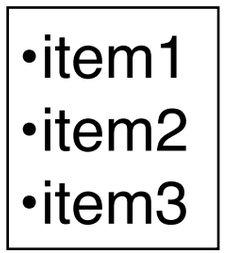
EVENT DELEGATION

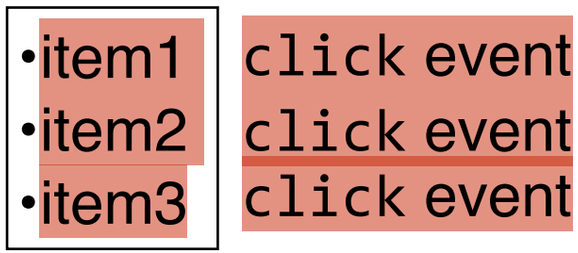
WITHOUT EVENT DELEGATION

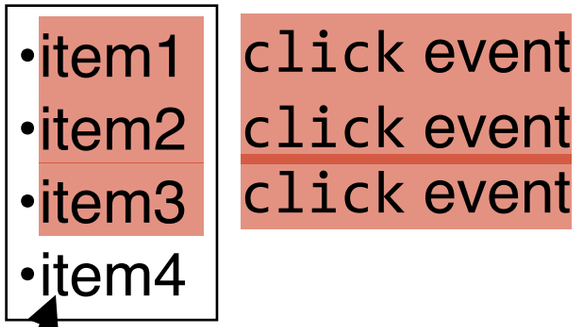
add an event listener to each li in the DOM

1. load page
2. set event listener on list items
3. add a new list item

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







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 from 'li' to 'ul'

new argument 'li' added to on() method

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

add an event listener to the ul element that applies to all of its li descendants

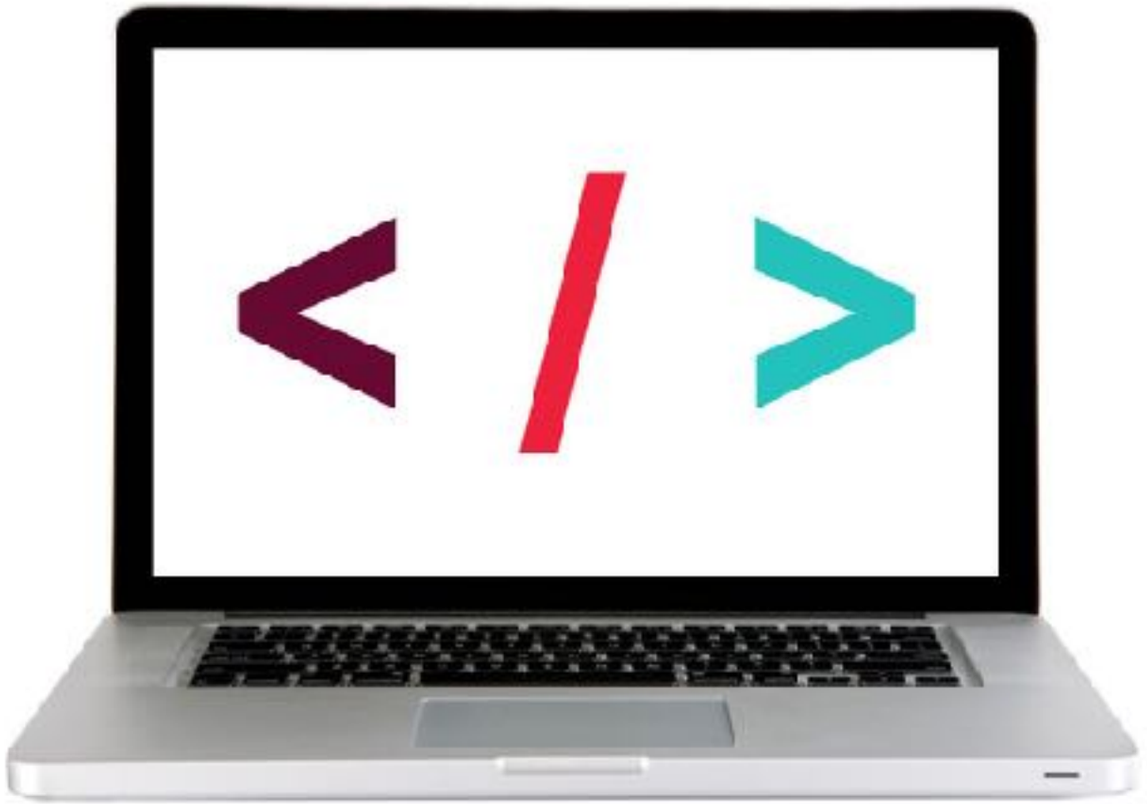
- item1 — click event
- item2 — click event
- item3 — click event

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!

INTRO TO JQUERY



LET'S TAKE A CLOSER LOOK

EXERCISE - EVENT DELEGATION

OBJECTIVE

- Use event delegation to manage dynamic content.

LOCATION

- `starter-code > 1-best-practices-exercise`

TIMING

10 min

1. Return to main.js in your editor and complete items 4a and 4b.
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: When the user mouses over each item, the item should turn grey. Don't use CSS hovering for this.
4. BONUS: 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

- Grouping all the events for an element in a single `.on()` statement makes our code more organized and is faster

```
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');
  }
});
```

EXERCISE - ATTACHING MULTIPLE EVENTS

LOCATION

▸ `starter-code > 2-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.

AJAX & APIS

ACTIVITY

TYPE OF EXERCISE

- Individual/Partner

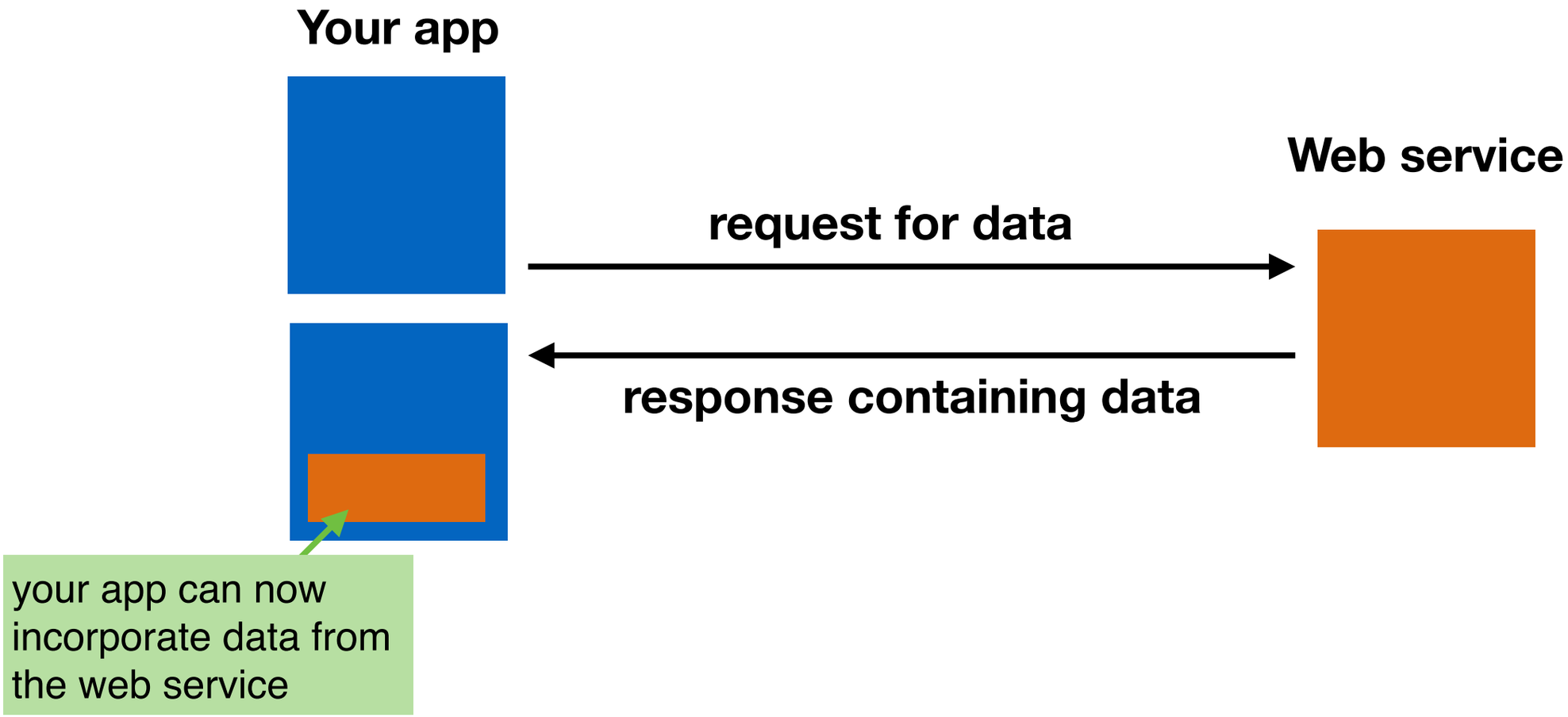
TIMING

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.

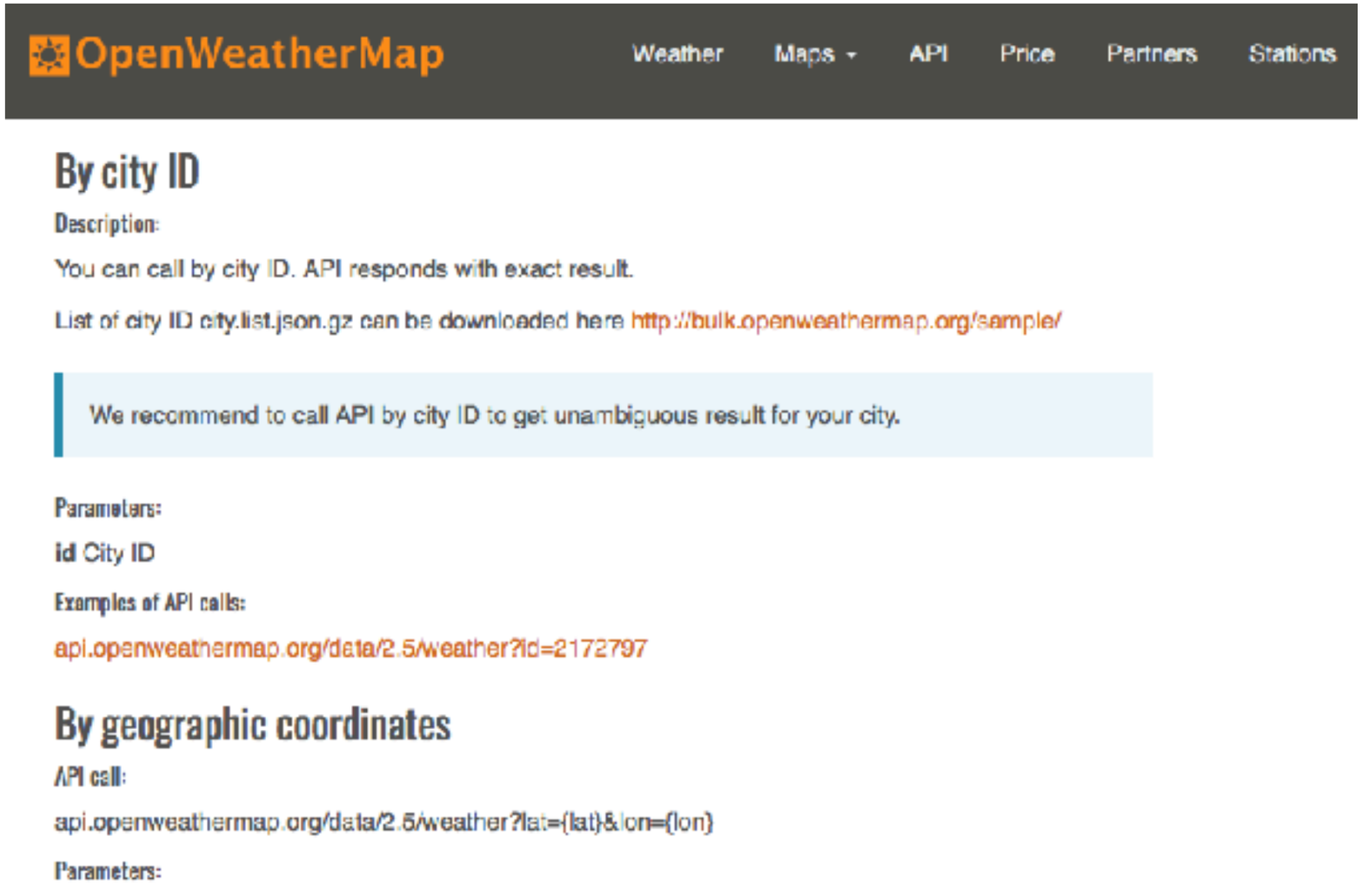
APIs

WEB SERVICES



WEB SERVICES

API = application programming interface



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)




WEB SERVICES

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:

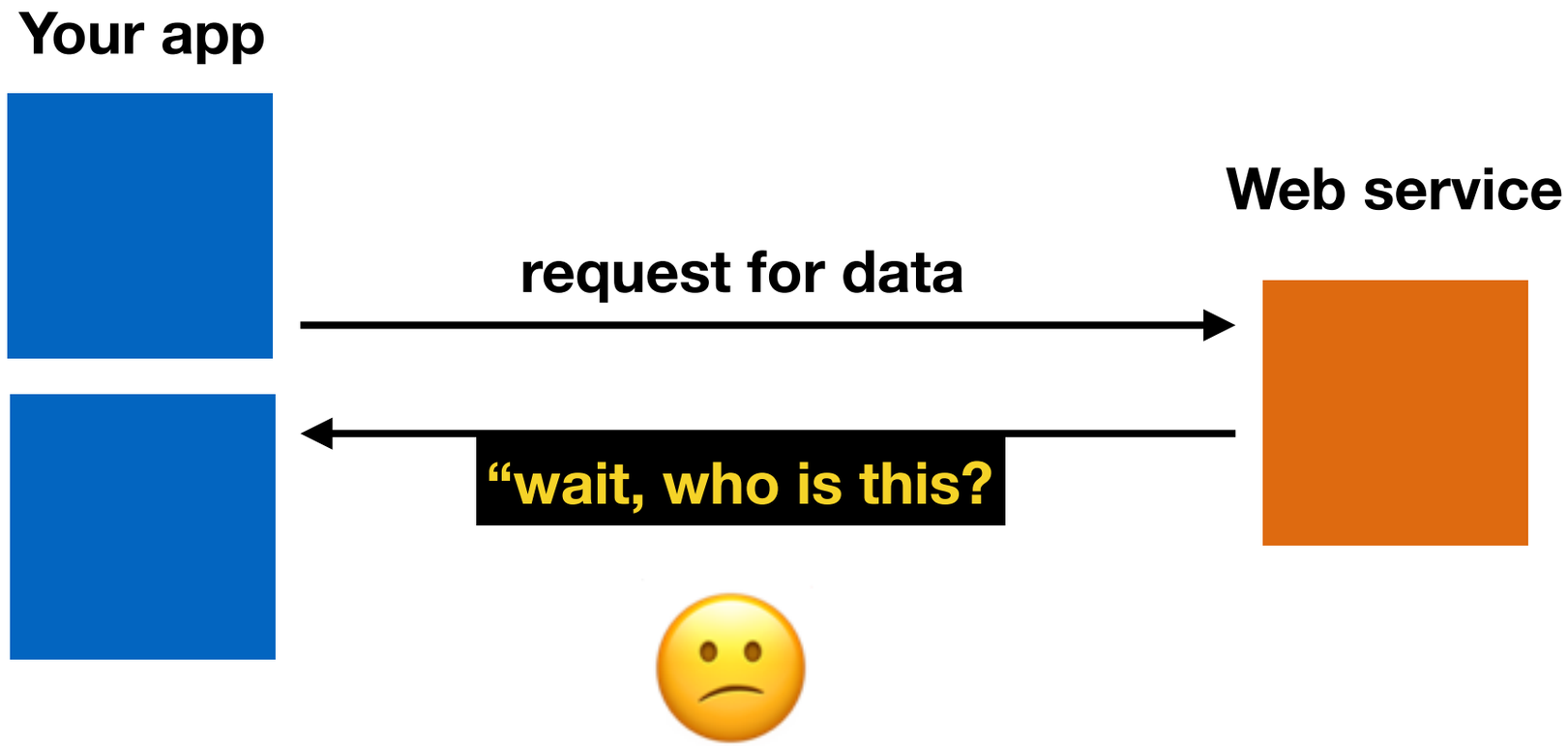
api.openweathermap.org/data/2.5/weather?lat=35&lon=139

API respond:

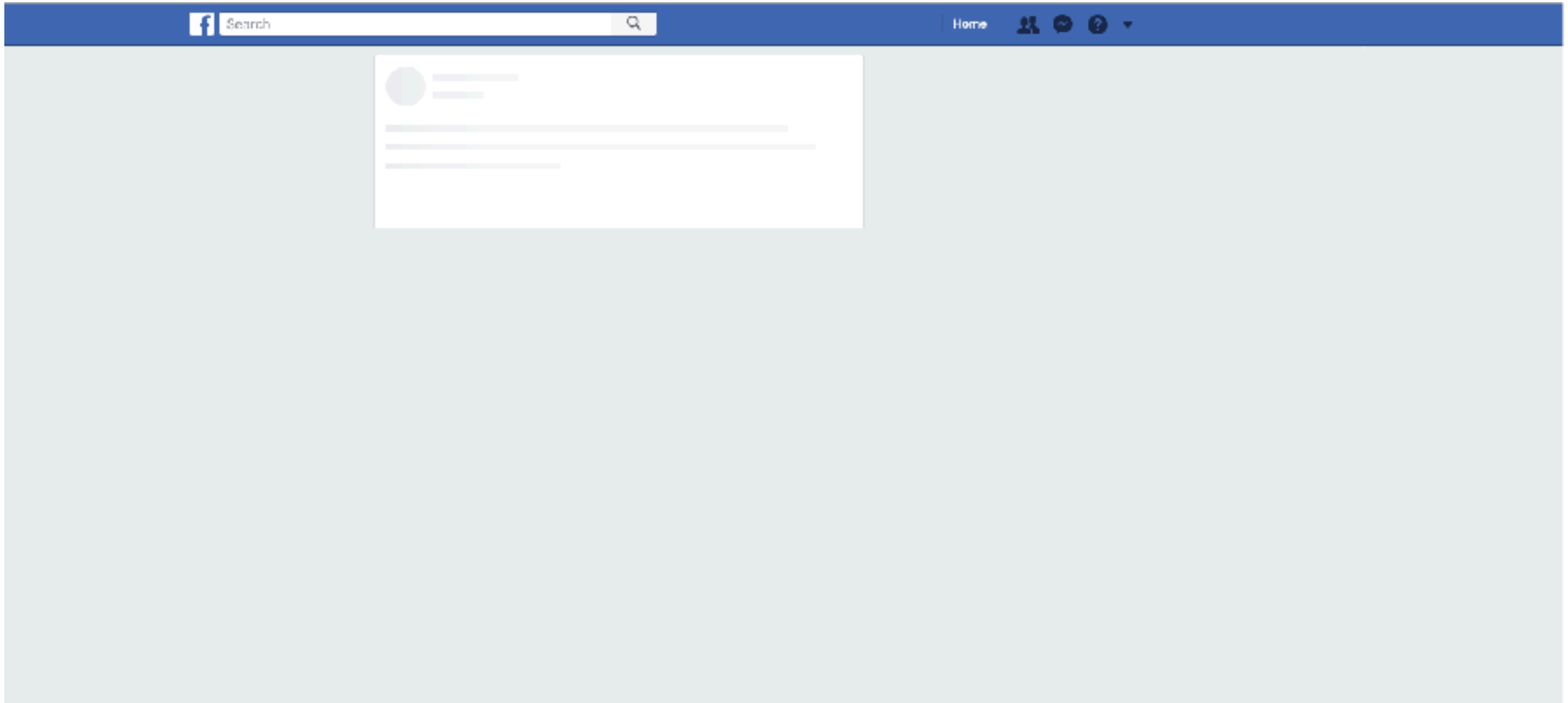

```
{"coord":{"lon":139,"lat":35},
"sys":{"country":"JP","sunrise":1369769524,"sunset":1369821049},
"weather":[{"id":804,"main":"clouds","description":"overcast clouds","icon":"0
4n"}],
"main":{"temp":289.5,"humidity":89,"pressure":1013,"temp_min":287.04,"temp_max
":292.04},
"wind":{"speed":7.31,"deg":187.002},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

WHAT WE NEED TO KNOW TO USE AN API

AN API MIGHT REQUIRE AUTHENTICATION



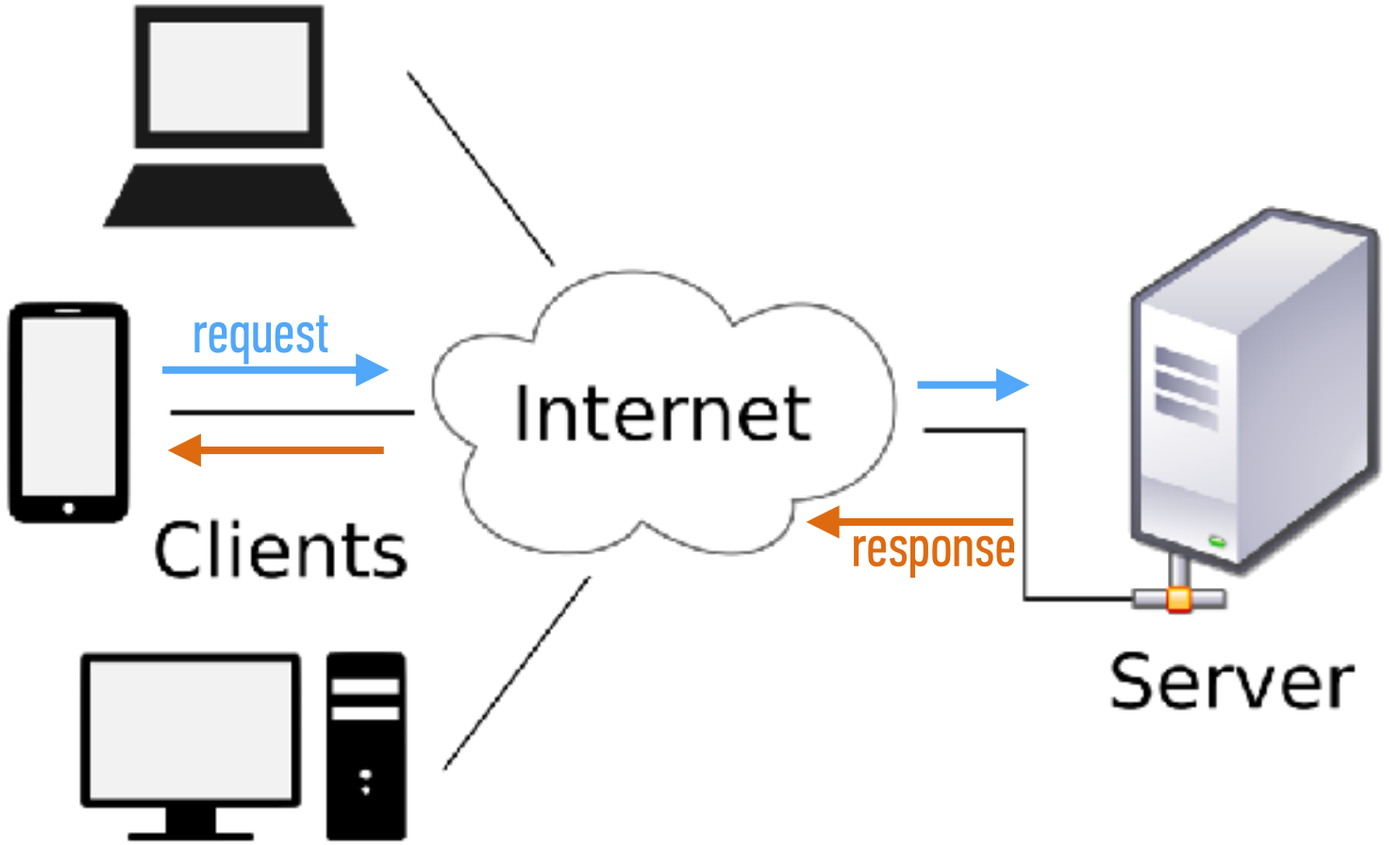
YOUR APP MIGHT EXPERIENCE A DELAYED RESPONSE



YOUR REQUEST MAY RESULT IN AN ERROR

REST (representational state transfer)

- architectural style of web applications
- transfers a representation of the state of a server resource to 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

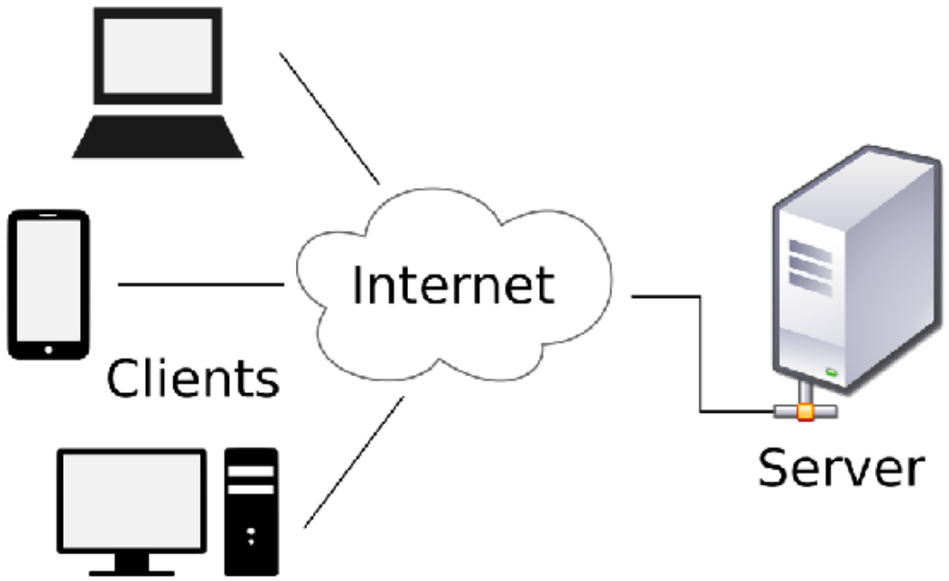
API respond:


```
{"coord":{"lon":139,"lat":35},
"sys":{"country":"JP","sunrise":1369769524,"sunset":1369821049},
"weather":[{"id":804,"main":"clouds","description":"overcast clouds","icon":"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}
```

HTTP

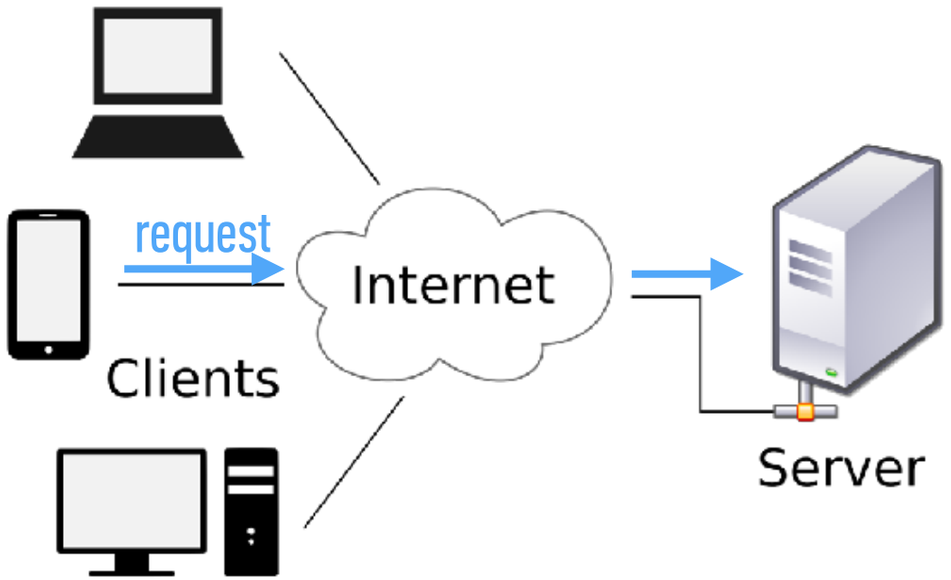
HTTP (hypertext transfer protocol)

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



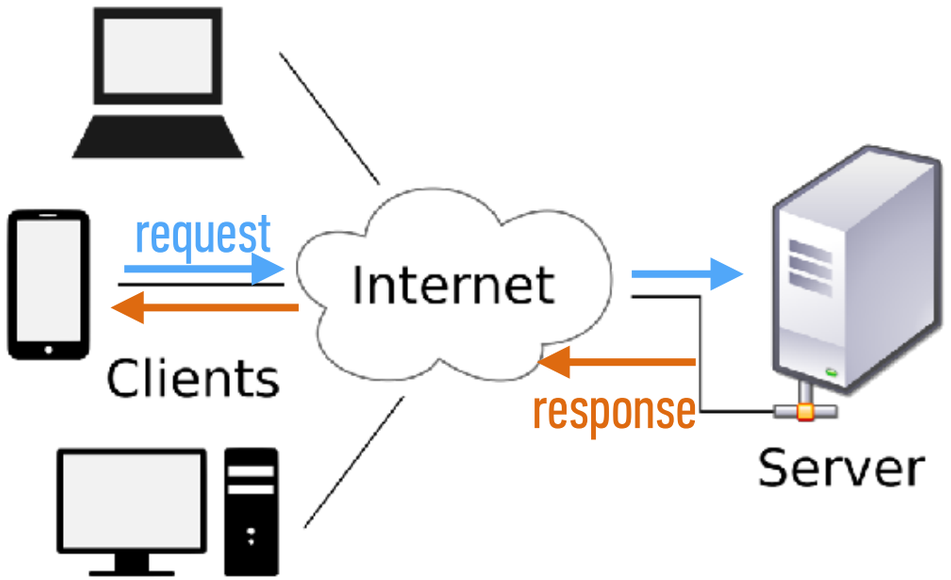
HTTP (hypertext transfer protocol)

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

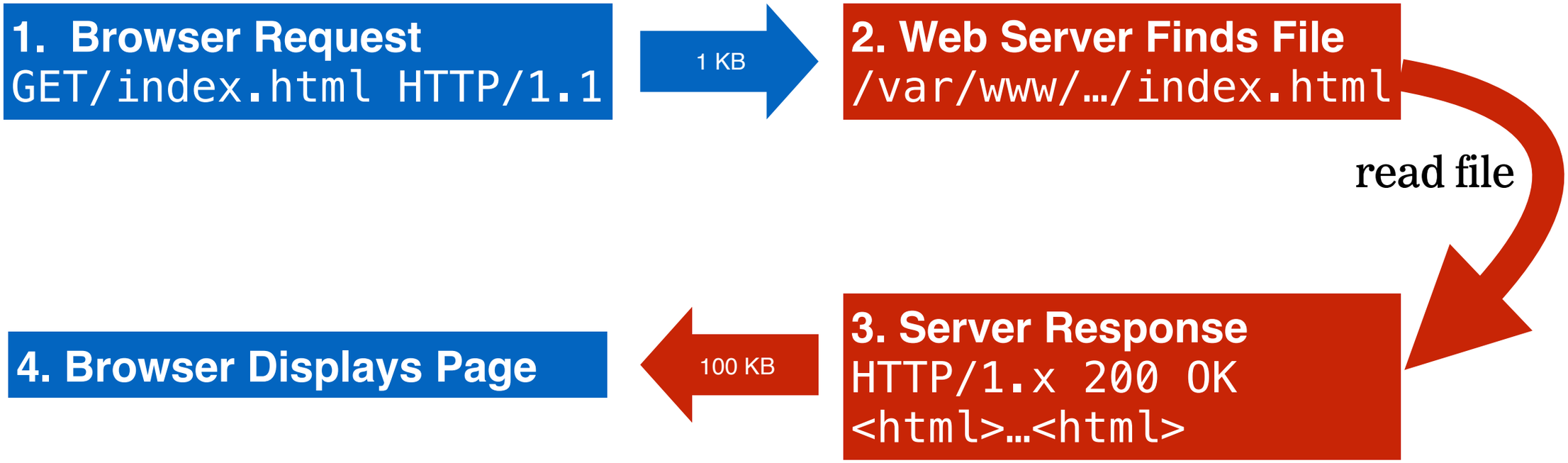


HTTP (hypertext transfer protocol)

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



HTTP REQUEST AND RESPONSE



HTTP (hypertext transfer protocol)

HTTP client

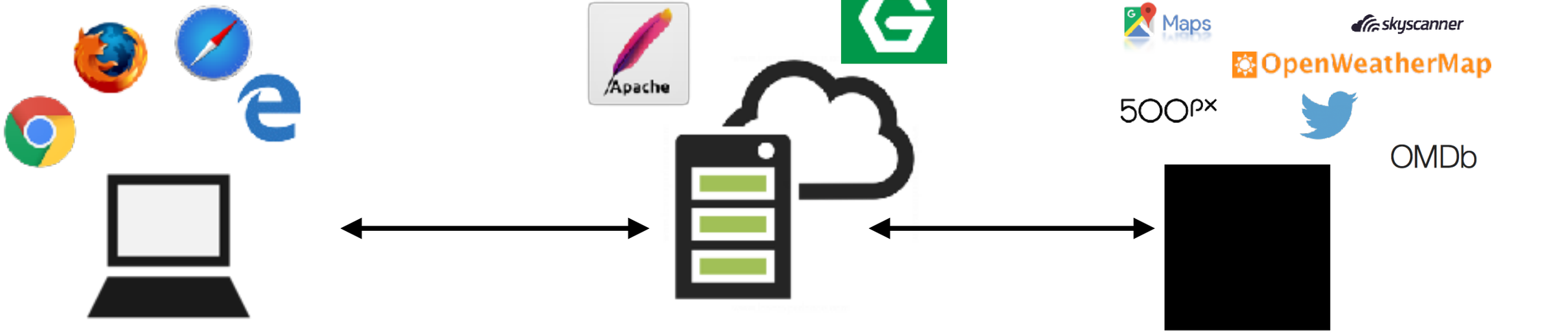
web browser

HTTP server

web server software

Web service

app that responds to HTTP requests

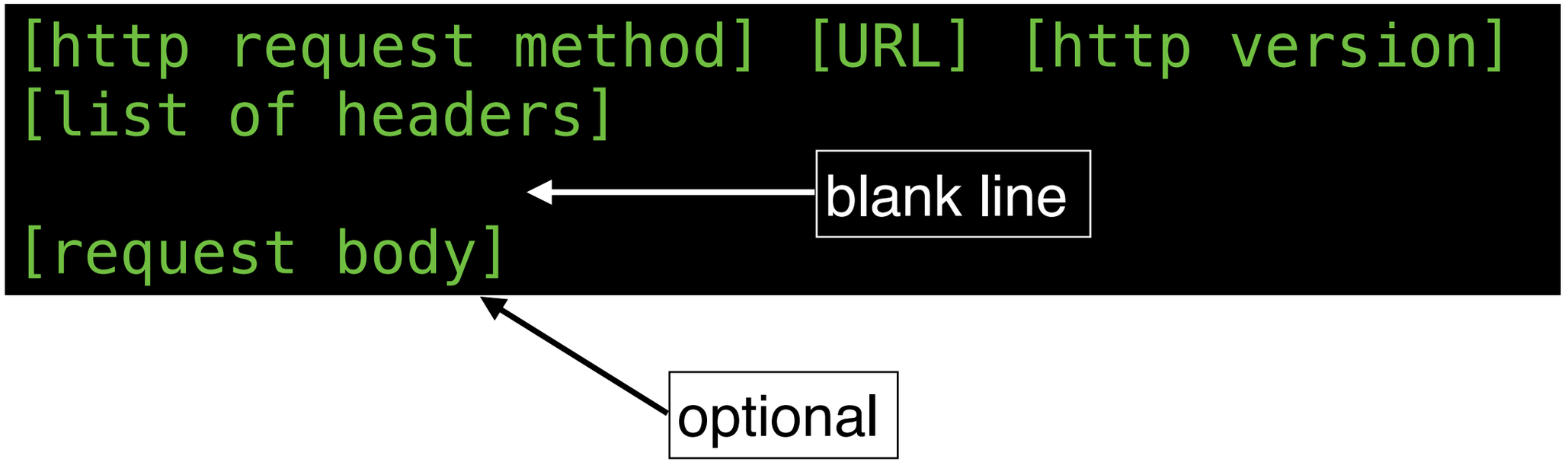


HTTP REQUESTS IN EVERYDAY LIFE

protocol host resource path query

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

HTTP REQUEST STRUCTURE

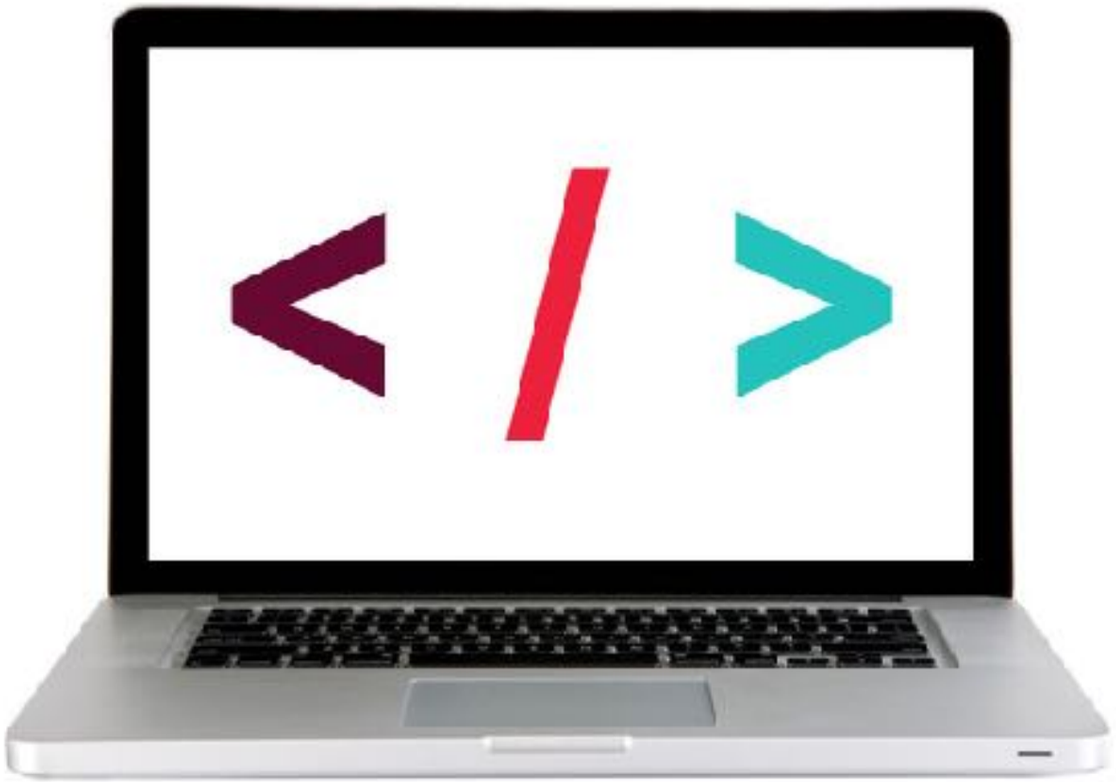


HTTP REQUEST METHODS ("HTTP VERBS")

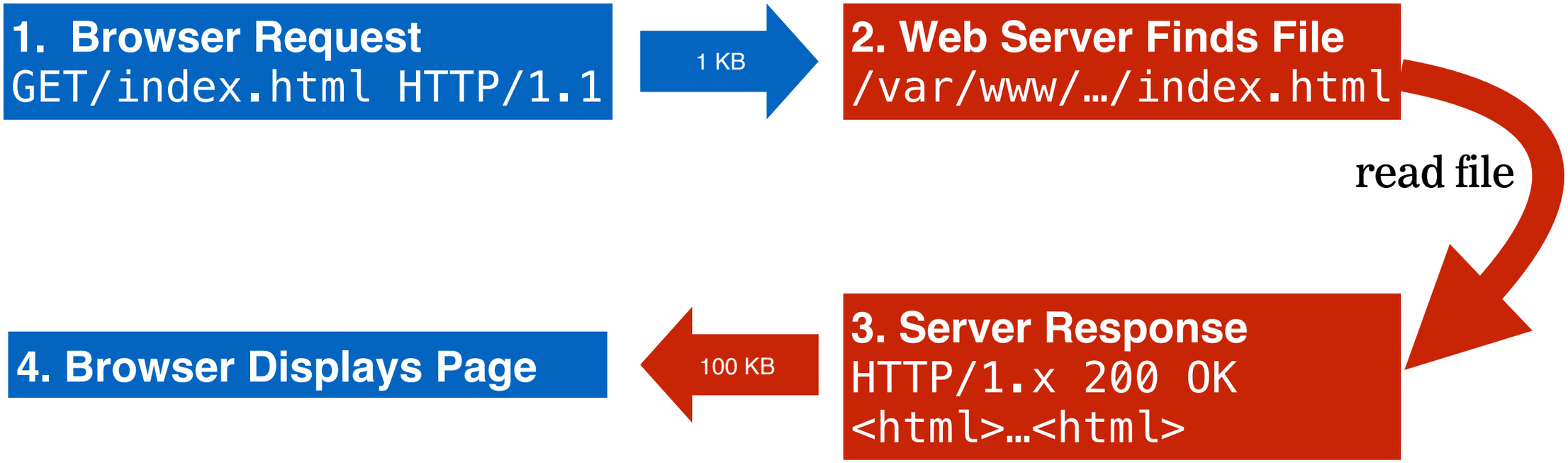
`GET`	Retrieve a resource
`POST`	Create a resource
`PATCH`	Update an existing resource (use instead of `PUT`, which replaces)
`DELETE`	Delete a resource
`HEAD`	Retrieve the headers for a resource

Most widely used

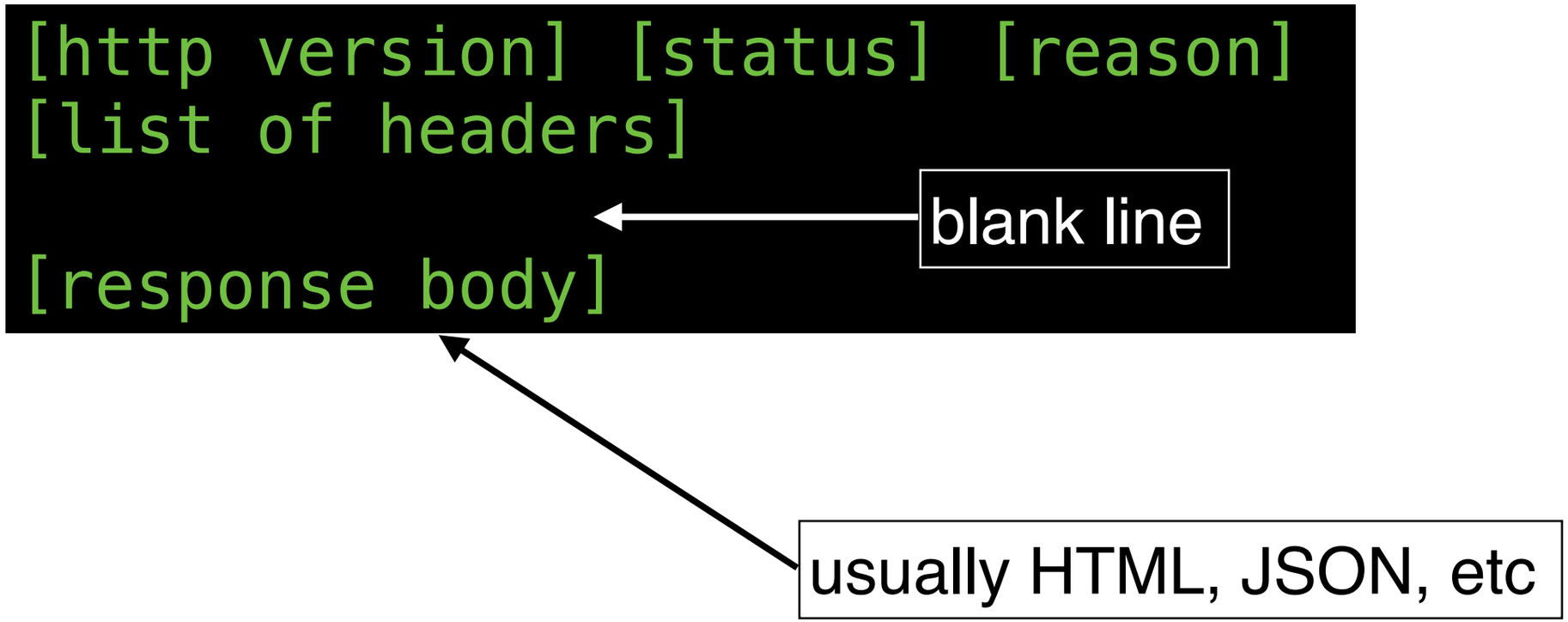
LET'S TAKE A CLOSER LOOK



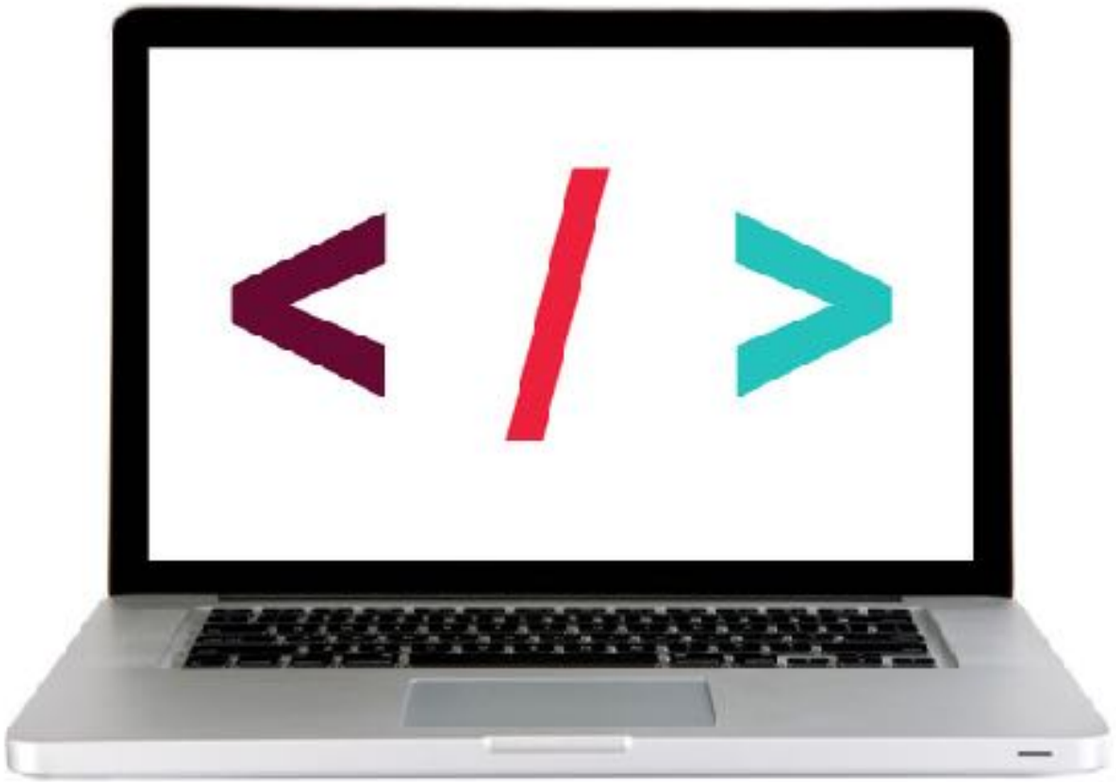
HTTP REQUEST AND RESPONSE



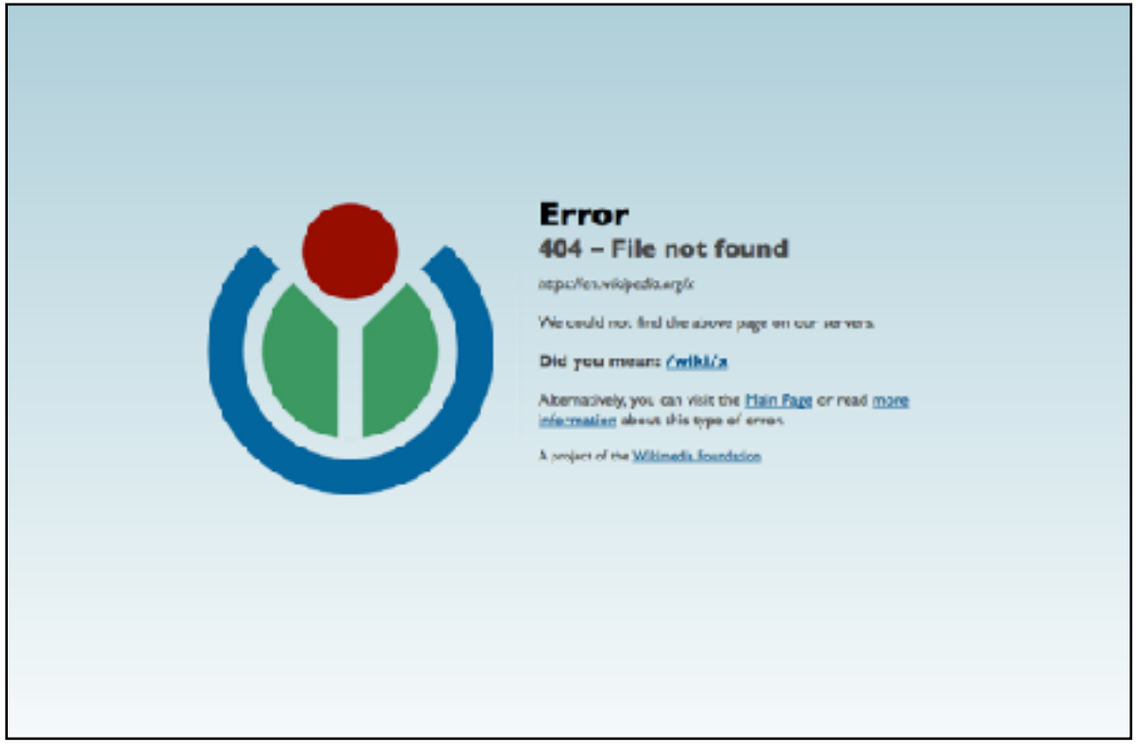
HTTP RESPONSE STRUCTURE



LET'S TAKE A CLOSER LOOK



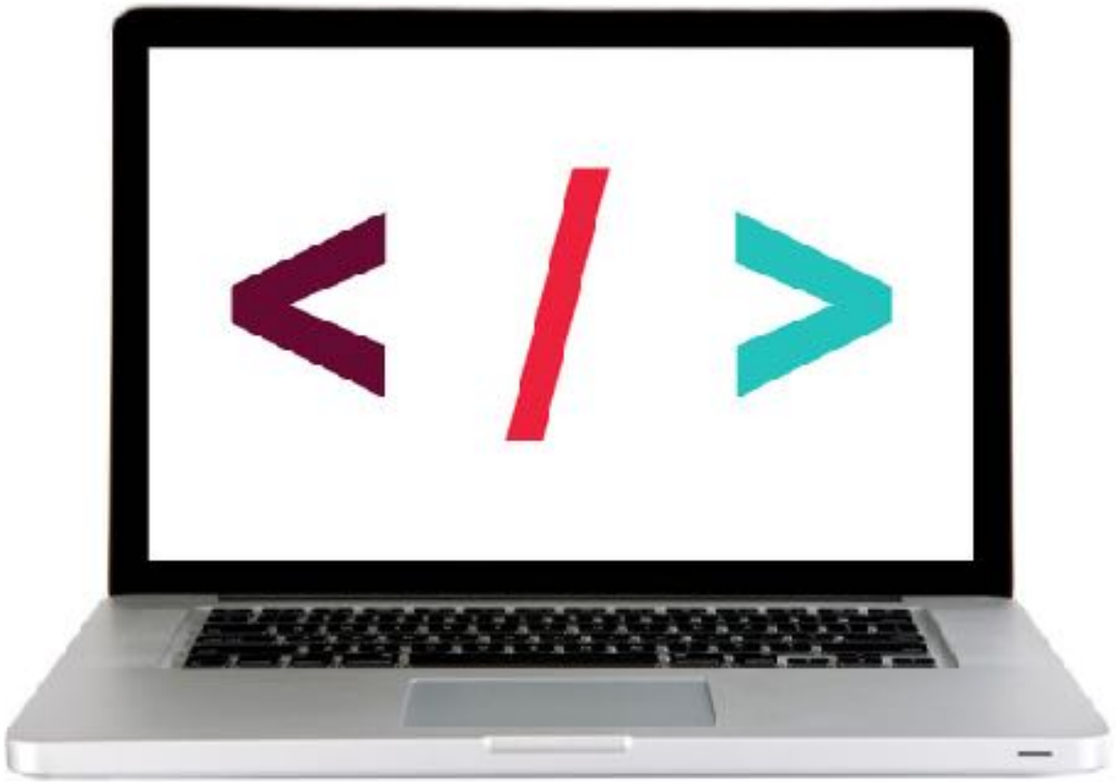
HTTP STATUS CODES



HTTP STATUS CODES

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

LET'S TAKE A CLOSER LOOK



Ajax

Ajax

A synchronous

J avaScript

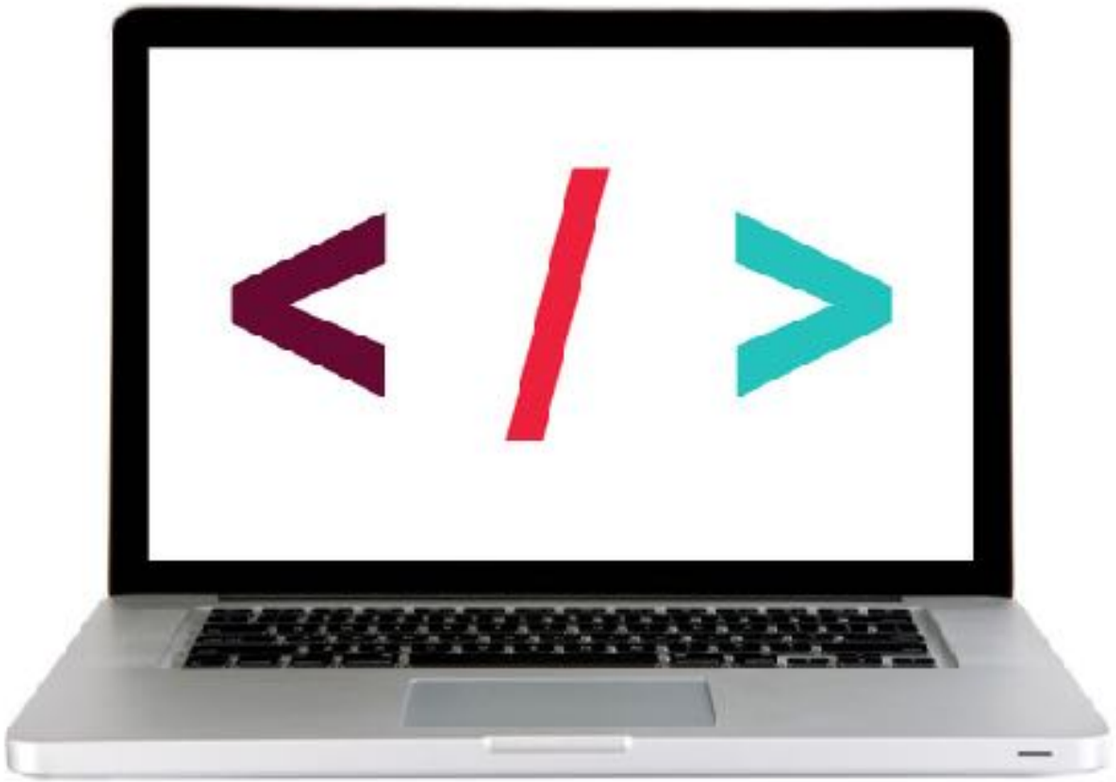
A nd

X ML **or JSON!**

Fetch = Ajax requests in vanilla JavaScript

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

LET'S TAKE A CLOSER LOOK



EXERCISE - CREATING AN AJAX REQUEST

LOCATION

- `starter-code > 4-ajax-exercise`

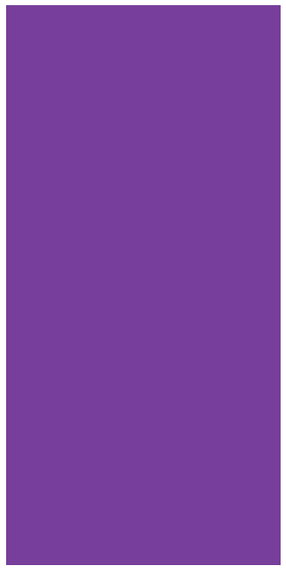
TIMING

5 min

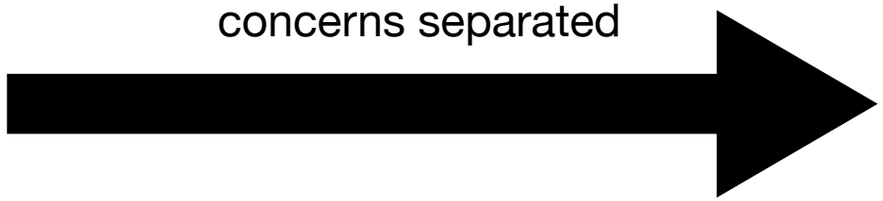
1. Copy the code from the codealong to the main.js file.
2. Change the URL to the one shown in the instructions.
3. Verify that a new set of results is shown in the console.
4. BONUS: Customize the error message to display the text of the HTTP status message.
   (Hint: look at https://developer.mozilla.org/en-US/docs/Web/API/Response/statusText)
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.

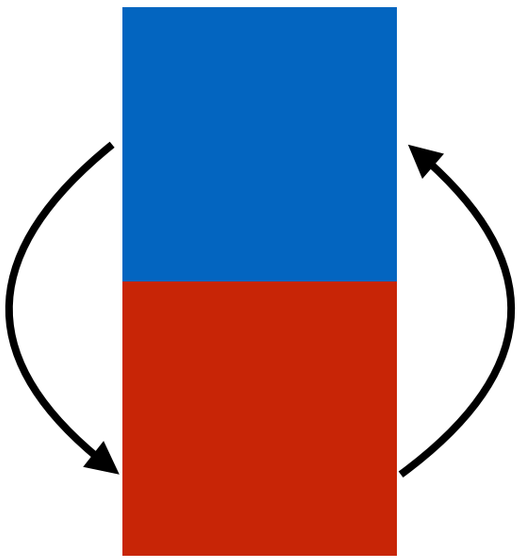
SEPARATION OF CONCERNS

code for data and view intermingled



concerns separated

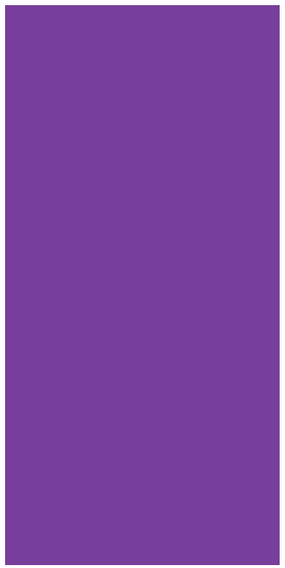




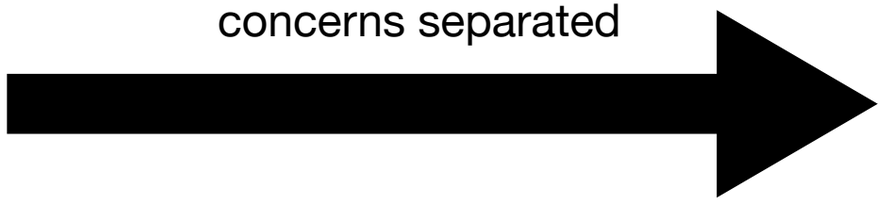
parts of code call each other, but are maintained separately

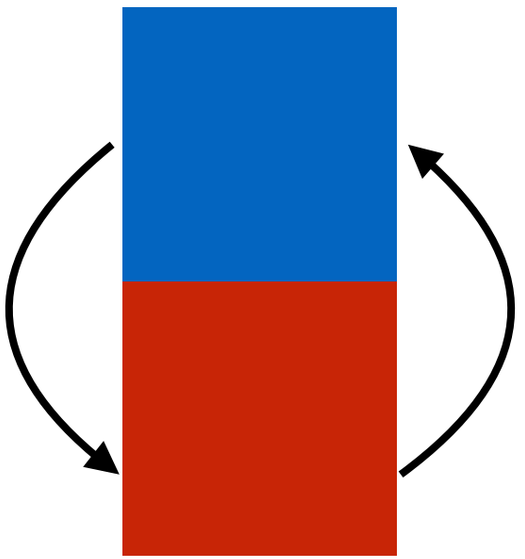
SEPARATION OF CONCERNS - HTTP

code for client and for HTTP requests intermingled



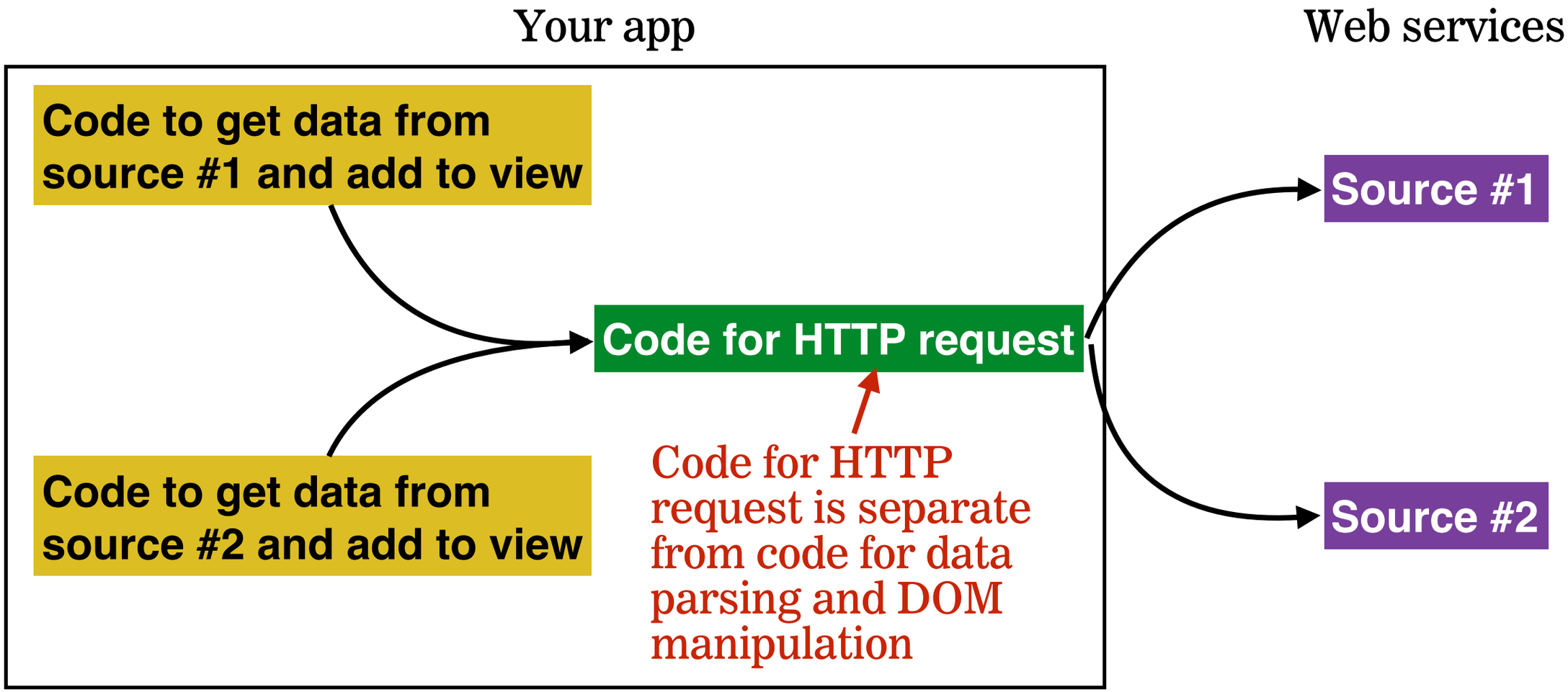
concerns separated



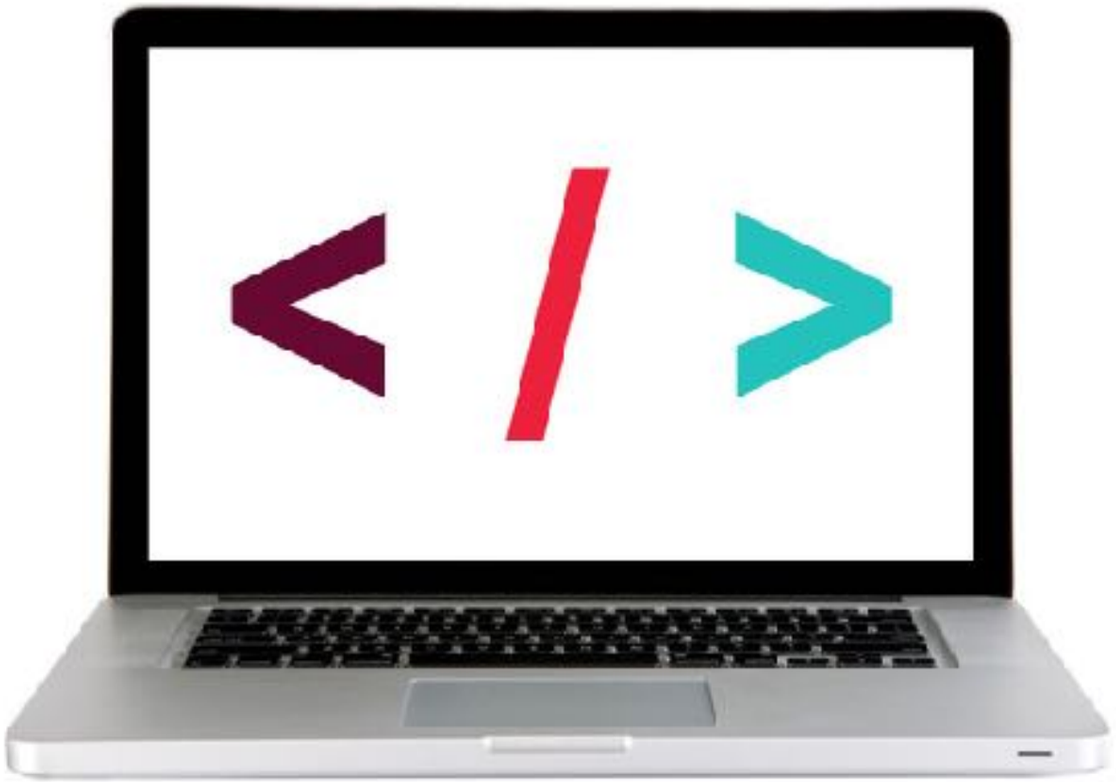


parts of code call each other, but are maintained separately

SEPARATION OF CONCERNS - HTTP



LET'S TAKE A CLOSER LOOK



EXERCISE – SEPARATION OF CONCERNS

TYPE

- Pairs

TIMING

2 min

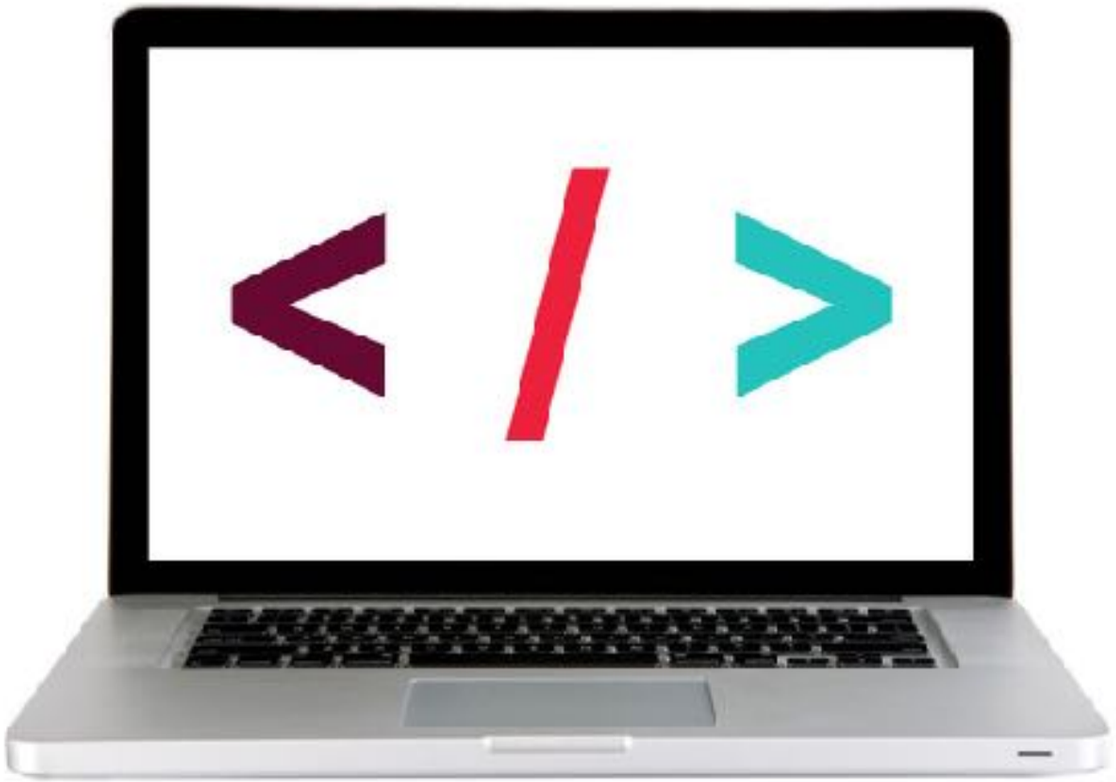
1. Imagine you're creating an app that displays the current weather from weather.com, and world news headlines from the LA Times
2. Spend 30 seconds thinking about how you might architect this app to implement separation of concerns; feel free to draw on your desk if that's helpful
3. After 30 seconds, find a partner and share your ideas for app architecture. Make note of what's similar and what's different in your plans.

jQuery Ajax

Using Ajax with jQuery

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

LET'S TAKE A CLOSER LOOK



LAB

LAB — JQUERY AJAX & SEPARATION OF CONCERNS

OBJECTIVES

- Implement a jQuery Ajax client for a simple REST service.
- Reiterate the benefits of separation of concerns – API vs. Client.

LOCATION

- `starter-code > Homework-5 > 7-jquery-ajax-exercise`

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 #10)

LEARNING OBJECTIVES - REVIEW

- Use event delegation to manage dynamic content in jQuery.
- 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.

Q&A