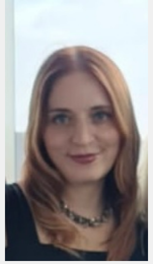
COM3504/6504 The Intelligent Web

Lecture 4: HTTP requests



Dr Vitaveska Lanfranchi
v.lanfranchi@sheffield.ac.uk
https://www.sheffield.ac.uk/dcs/people/academic/vitaveska-lanfranchi

Mr. Andy Stratton
a.stratton@sheffield.ac.uk
https://www.sheffield.ac.uk/dcs/people/academic/andrew-stratton

Dr Fatima Maikore
F.Maikore@sheffield.ac.uk
https://www.sheffield.ac.uk/dcs/people/research-staff/fatima-maikore

Let’s start

- Why real-time communication?
- What are the main technologies involved
  - Data Formats for interchange
    - JSON
  - HTPP requests

Learning Objectives

- During this unit you will learn about:
  - What JSON is

    Why it is important

    How we can use JSON with Node.JS
  - How do we query a server/API?

    Axios

    Ajax

    Fetch

JSON

What is JSON?

- JSON stands for **J**ava**S**cript **O**bject **N**otation
  - It is a standard for storing, transmitting and processing data

    Data interchanges language
  - Often used to communicate data between a web server and a client
  - Is text only
  - Very easy to understand for a human and for a machine
  - Completely language independent

    Can be used by any programming language.

JSON Data Structure

- JSON is built on two structures:
  - A collection of name/value pairs. In various languages, this is realized as an object, record, struct, dictionary, hash table, keyedlist, or associative array
  - An ordered list of values. In most languages, this is realized a an array, vector, list, or sequence.
- These are universal data structures.
  - all modern programming languages support them

JSON example

```
{
"students":[
    {"firstName":"Mark", "lastName":"Smith"},
    {"firstName":"Anna", "lastName":"Jones"},
    {"firstName":"Laura", "lastName":"Parker"}
]
}
```

JSON Syntax

- Data is in key/value pairs
  - Same notation as JavaScript objects
  - *keys* must be strings, written with commas
  - Values can be
    - a string
    - a number
    - an object (JSON object)
    - an array
    - a boolean
    - null

"firstName":"Mark"

JSON Syntax – cont.

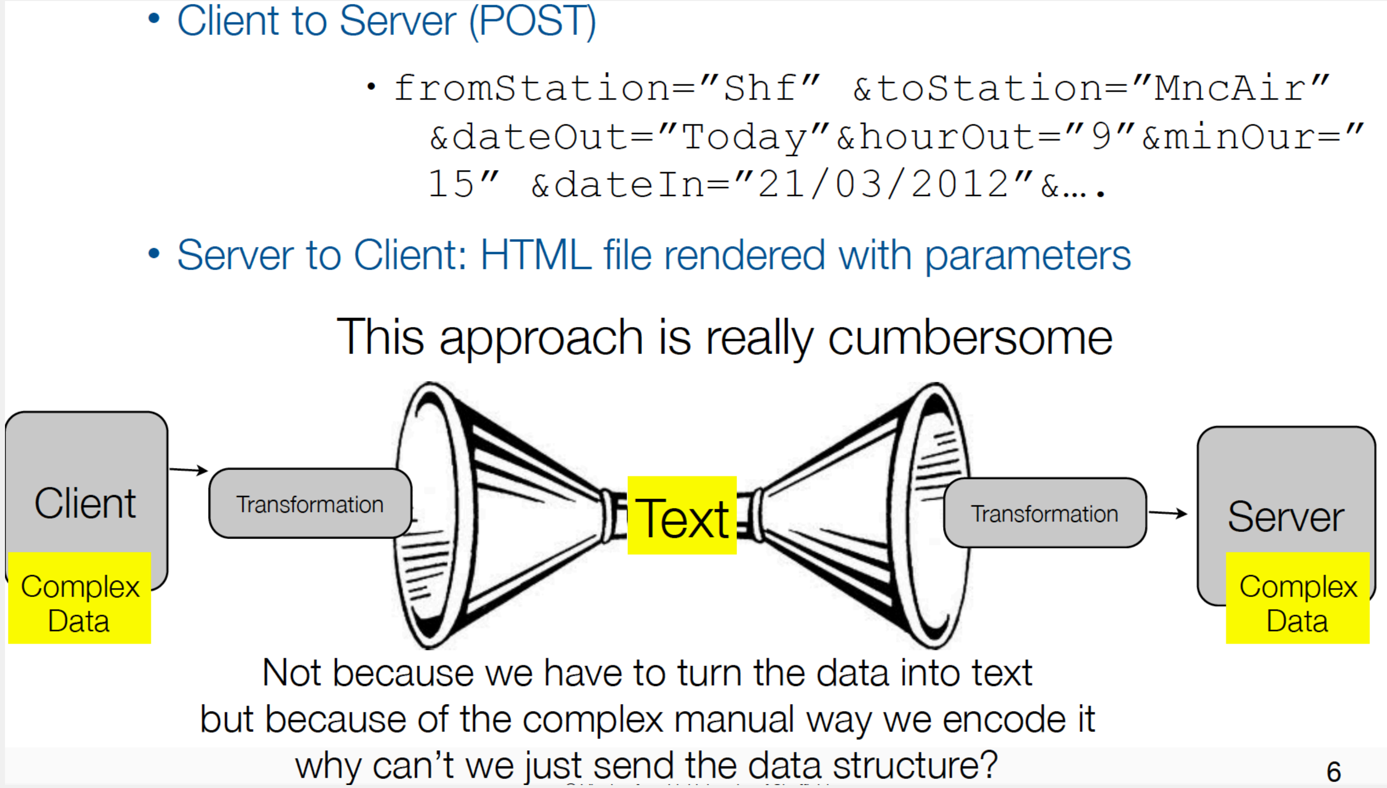
- Data is separated by commas
  - "firstName":"Anna", “lastName":"Jones"
- Curly brackets hold objects
  - {"firstName":"Mark", “lastName":"Smith"}
- Square brackets hold arrays
  - Remember an array can contain other arrays

JSON File and MIME type

- The file type for JSON files is “.json"
- The MIME type for JSON text is "application/json"

WHY IS JSON USEFUL FOR WEB ARCHITECTURES?

Traditional data serialisation/de-serialisation

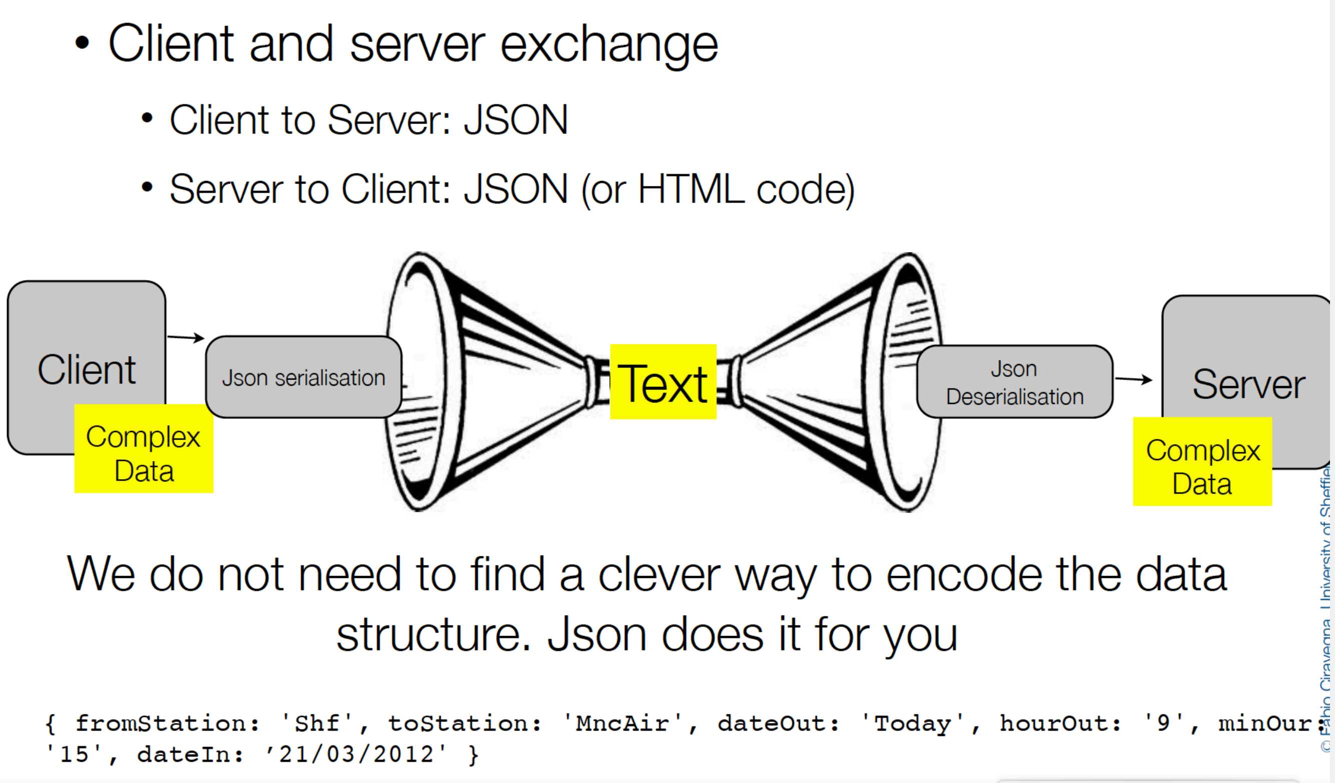


- Client to Server (POST)
  - `fromStation=”Shf” &toStation=”MncAir” &dateOut=”Today”&hourOut=”9”&minOur=”15” &dateIn=”21/03/2012”&….`
- Server to Client: HTML file rendered with parameters

This approach is really cumbersome

Not because we have to turn the data into text
but because of the complex manual way we encode it
why can’t we just send the data structure?

JSON Communication



- Client and server exchange
  - Client to Server: JSON
  - Server to Client: JSON (or HTML code)

We do not need to find a clever way to encode the data structure. Json does it for you

```
{ fromStation: 'Shf', toStation: 'MncAir', dateOut: 'Today', hourOut: '9', minOur:
'15', dateIn: ’21/03/2012' }
```

© Fabio Ciravegna, University of Sheffiel

JSON vs XML

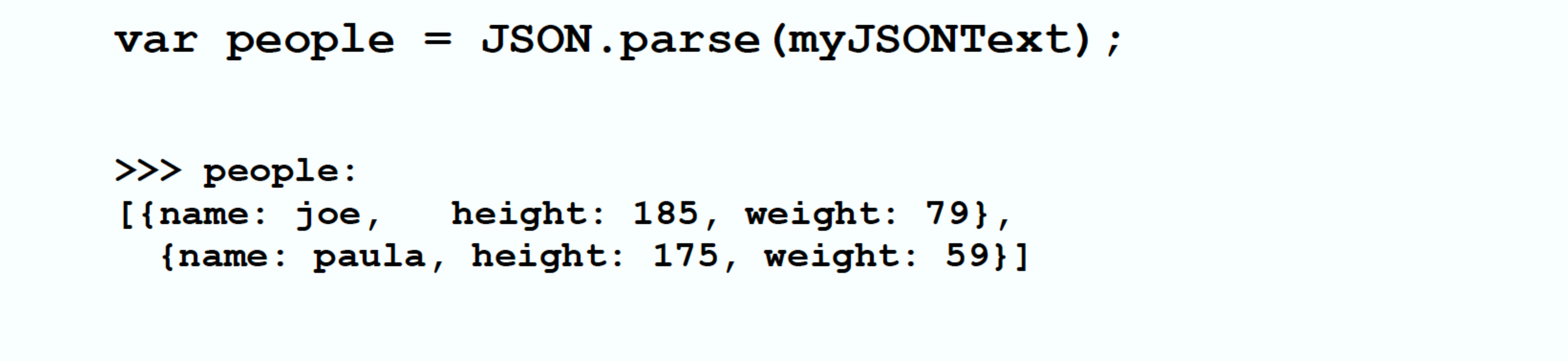
- Why JSON and not XML?
  - XML needs an XML parser to deal with the content
  - XML does not support arrays
  - JSON can be parsed by a standard JavaScript function.

JSON AND NODE.JS

Parse an object into JSON with Node.JS

- Why is it easy to use with Node.JS?
  - As JSON uses JavaScript syntax a JavaScript program can easily convert JSON data into JavaScript objects.
- There is a pre-made JavaScript function that convert a JSON string into a Javascript object
  - part of the JavaScript standard since ECMAScript 5
  - After that you can process the object easily in your code
- JSON.parse()

JSON.parse example



```
var people = JSON.parse(myJSONText);


>>> people:
[{name: joe,   height: 185, weight: 79},
  {name: paula, height: 175, weight: 59}]
```

Send data to a server using JSON

- If you want to send data to a server that accepts JSON
  - Convert it into a Javascript object
  - Use JSON.stringify()

Example

- Given the code

```
var obj = { name: "Mark", age: 20, city: "Sheffield" };
var myJSON = JSON.stringify(obj);
```

- The output is

```
{“name":"Mark","age":20,"city":"Sheffield"}
```

- You can also stringify arrays

HTTP REQUESTS

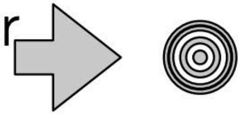
Traditional client server communication

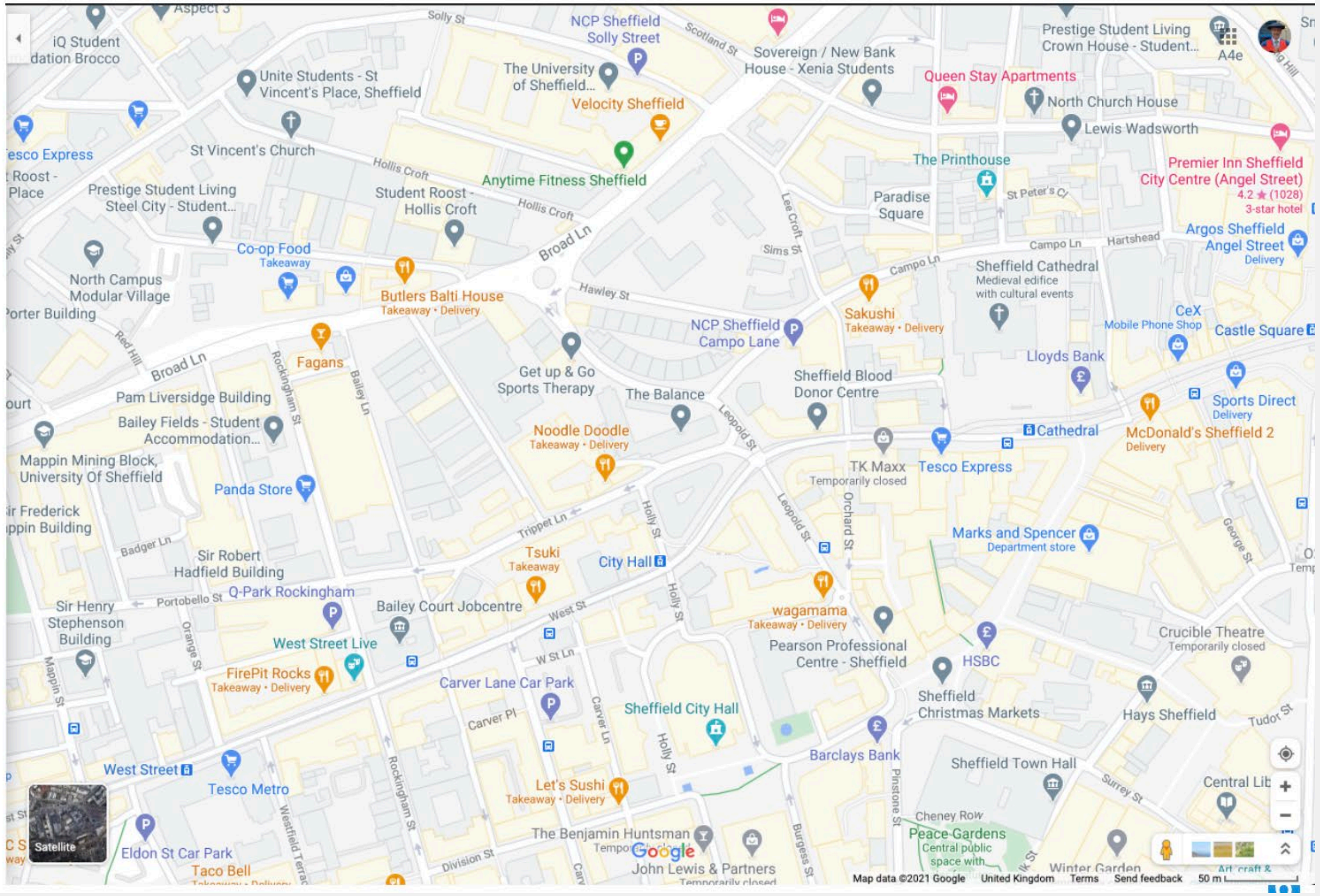
- Traditional communication implies sending a request (GET/POST) and receiving a full document to Display (e.g. HTML+CSS)
- Communication is stateless and directional
- The next request will have to contain all the information because the server will have forgotten everything about my request

Example: Mapping

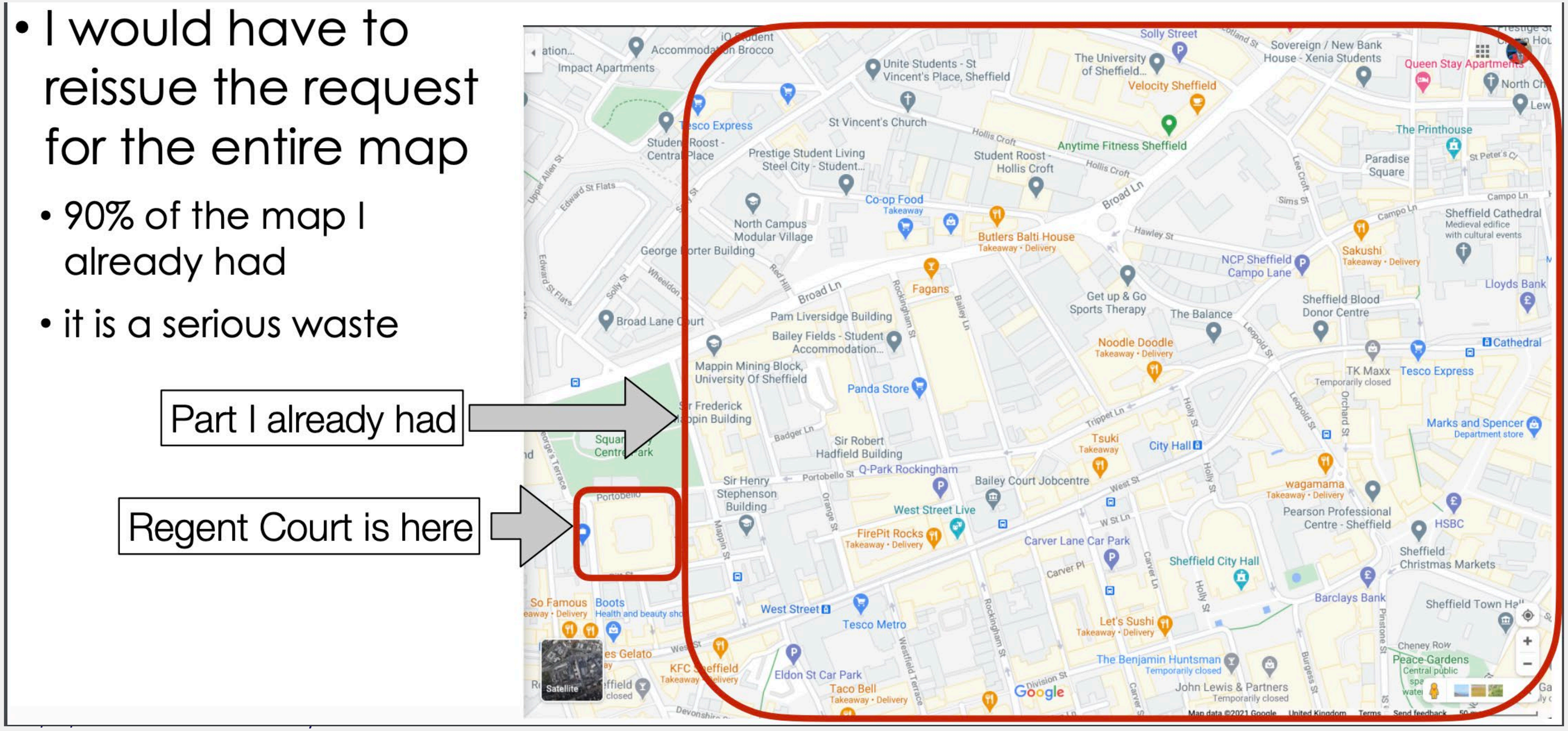
- Suppose I have this map. Regent Court is not visible
  - it is just outside

it is more or less here





Example: Mapping 2



- I would have to reissue the request for the entire map
  - 90% of the map I already had
  - it is a serious waste

Part I already had

Regent Court is here

A new type of communication

- We would like the client
  - to be able to request just the missing tiles of the map and reuse those that are already there
  - to allow to use the map while the missing tiles are fetched
- Advantages:
  - Less traffic
  - The browser page would not freeze while the data is fetched
  - Better user experience

HTTP requests

- Retrieving or modifying API data from a server is a vital part of most web applications.
  - loading user information,
  - receiving updates from the server

How can we achieve it?

- Three main popular methods
  - Ajax

    a Javascript middleware built in in the browser which intercepts the requests to the server and may help optimising them

    being event based, while communication is ongoing, the browser is still alive and usable by the user
  - Axios
  - Fetch

Main differences

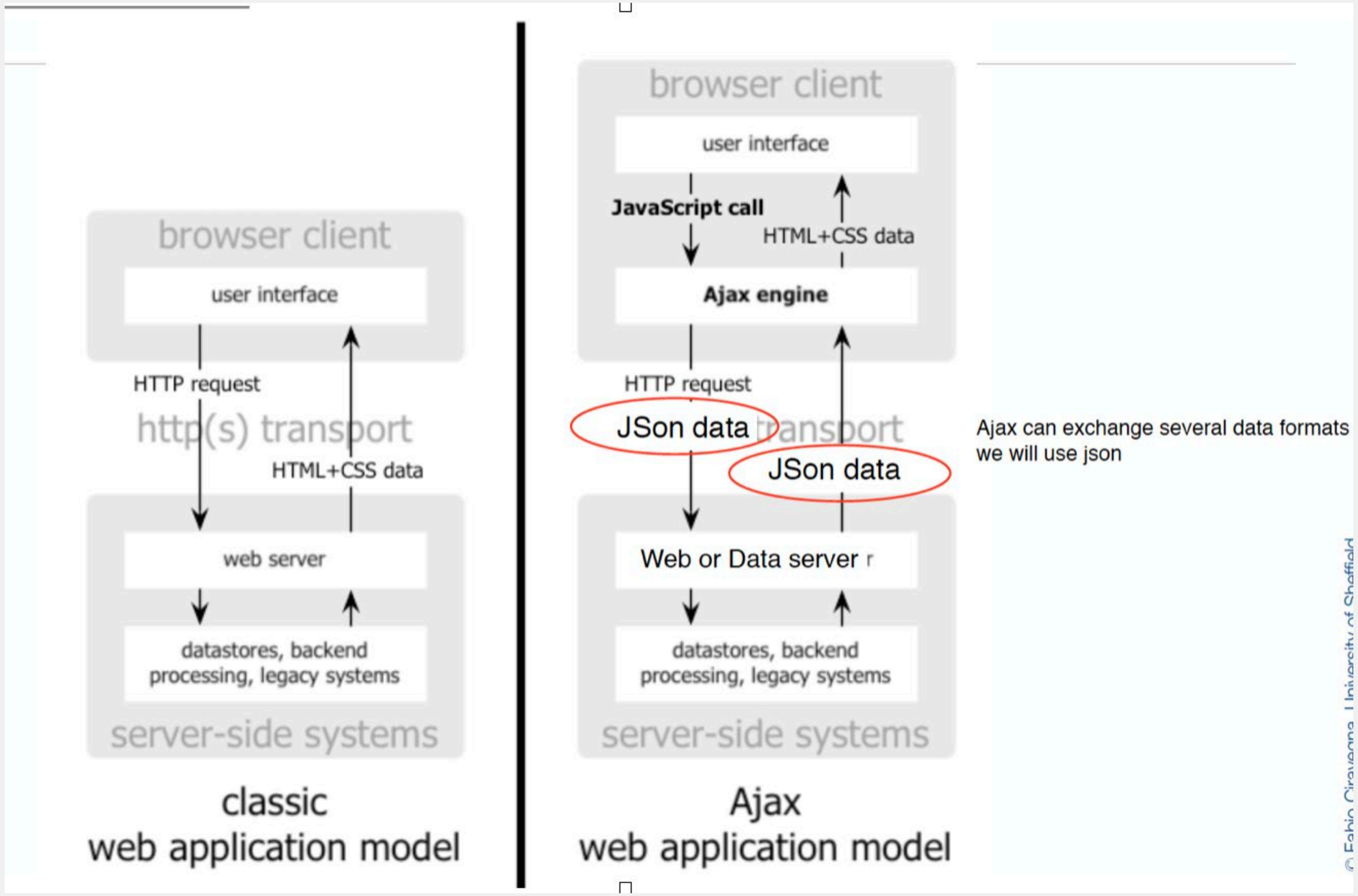
- Ajax is a concept
  - You do not have to reload a whole HTML page to update its content
- First Ajax implementation was using XMLHttpRequest
  - standard way to do HTTP requests from JavaScript to get content and update the HTML page with it.
- Axios is not a replacement of Ajax, it facilitates the way to build “Ajax-based” web applications

AJAX

What is Ajax

- Ajax is a library for Asynchronous client-server communication
- The Ajax engine built in in the browser will
  - render the user interface
  - communicate with the server on the user's behalf.
- The Ajax engine allows the user's interaction with the application to happen asynchronously
  - independent of communication with the server

Ajax Data Exchange



Ajax can exchange several data formats
we will use json

© Fabio Ciravegna, University of Sheffield

What can Ajax do?

- Every user action that normally would generate an HTTP request takes the form of a JavaScript call to the Ajax engine instead.
- Any response to a user action that doesn’t require the server will be handled by Ajax.
- If the engine needs something from the server in order to respond will make those requests asynchronously

Ajax Main Functionalities

- Update a web page without reloading it
- Request data from a server – after the page has loaded
- Receive data from a server – after the page has loaded
- Send data to a server in background
- Ajax uses a combination of
  - A browser built-in XMLHttpRequest object (to request data)
  - Javascript + HTML DOM to display the data

How does Ajax work?

- Ajax uses a programming model with display and events.
- These events are user actions, they call functions associated to elements of the web page.
- Interactivity is achieved with forms and buttons.
- The original AJAX way of programming is very complicated
- It normally uses JQuery which makes interaction very simple
  - See slides in Appendix if you want to try it

AXIOS

What is Axios

- third-party HTTP client library for making network requests
  - To your own server or external servers
- Returns a JSON object
- You simply need to import and require the axios module

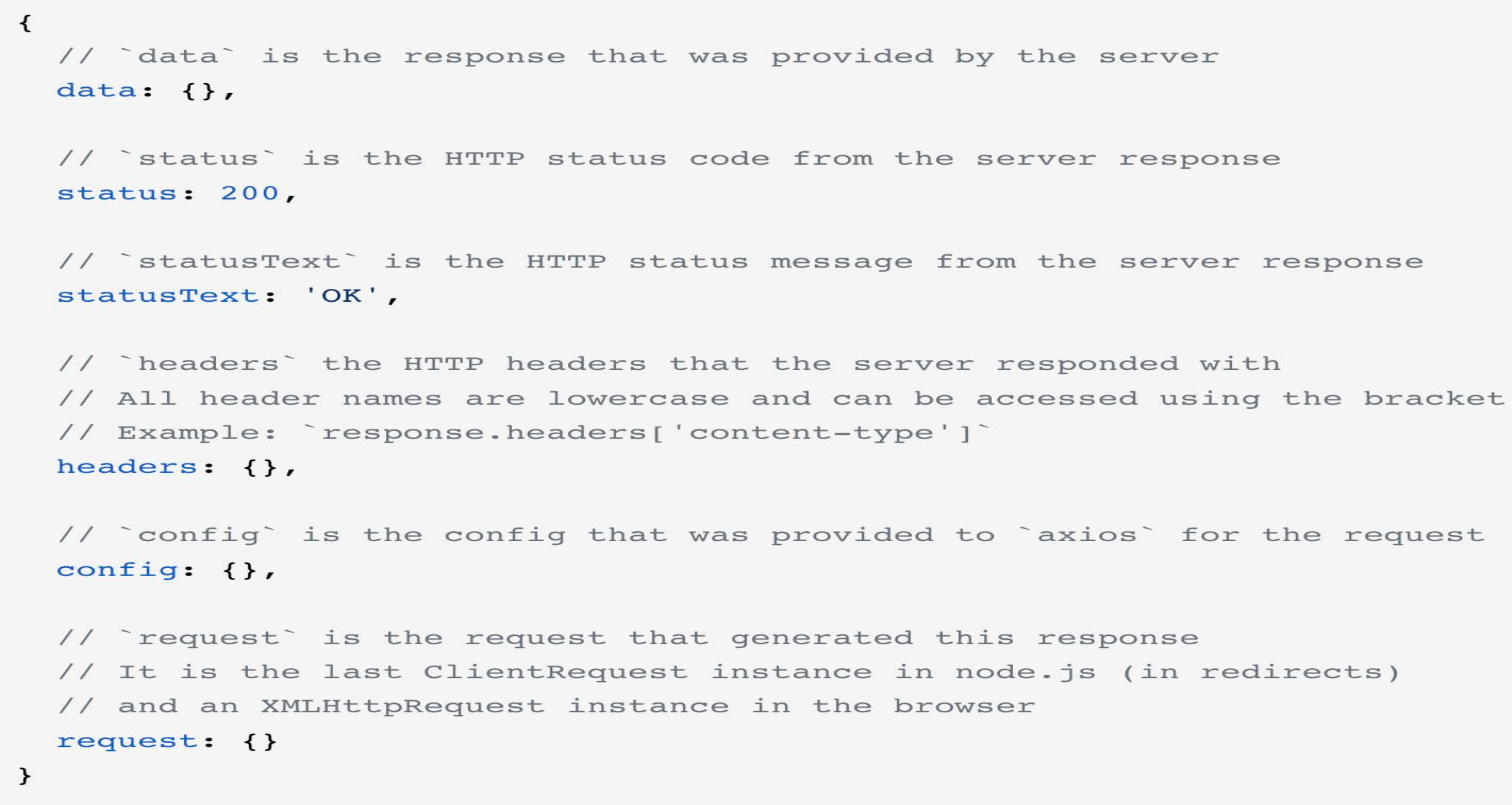
https://www.npmjs.com/package/axios

Axios get call

- Axios allows you to perform get requests
  - Youc an add headers like login details

```
axios .get(url)
    .then((response) => { displayOutput(response) })
    .catch((err) => console.log(err));
```

Response schema



```
{
  // `data` is the response that was provided by the server
  data: {},

  // `status` is the HTTP status code from the server response
  status: 200,

  // `statusText` is the HTTP status message from the server response
  statusText: 'OK',

  // `headers` the HTTP headers that the server responded with
  // All header names are lowercase and can be accessed using the bracket
  // Example: `response.headers['content-type']`
  headers: {},

  // `config` is the config that was provided to `axios` for the request
  config: {},

  // `request` is the request that generated this response
  // It is the last ClientRequest instance in node.js (in redirects)
  // and an XMLHttpRequest instance in the browser
  request: {}
}
```

Example of Axios get

```
axios.get('https://go-apod.herokuapp.com/apod')
  .then((response) => {
    console.log(response.data);
    console.log(response.status);
    console.log(response.statusText);
    console.log(response.headers);
    console.log(response.config);
  });
```

Axios POST

- Axios.post allows to send post request attaching data
- You could use to upload images or files

```
axios.post(url,
        {id: 11, name: ”XXX", username: ”YYY", email:
”XXXYYY@gmail.com", }) .then((response) =>
displayOutput(response)) .catch((err) => console.log(err));
```

Axios interceptions

- Axios allows to intercept and manipulate HTTP requestsa and responses before they are handled by the then() or the catch() code block.
- Allows you to add details, customise the response etc

https://www.npmjs.com/package/axios#interceptors

FETCH

What is Fetch

- Promise-based HTTP client
- interface for fetching resources that allows to manipulate HTTP pipeline
  - Request and response are just one part of it
- Has a broader scope than Axios
- It’s built-in in recent versions of Node.JS
  - No need to import if you are using node >18
- It’s lower level, so might be more difficult to use

A warning

- If you run Node <18
- You must use node-fetch module
  - If you use "node-fetch": "~2.6.9” or earlier

    ```
    const fetch = require("node-fetch");
    ```

  - If you use "node-fetch": 3 or above

    You must import it - `import fetch from "node-fetch"`

https://blog.logrocket.com/es-modules-in-node-today/

Fetch Get

- The fetch() method returns a Promise object

```
fetch(URL)
  .then((response) => response.text())
  .then((body) => {
    res.send(body);
  });
```

- The first then() extracts the text from the response
- the second then() send back the response HTML

Headers

- You can use the second parameter of fetch() to send custom request headers
- The response object contains all of the response headers in the response.headers collection

```
fetch(url)
.then(response => { for(const pair of response.headers){ console.log(`${pair[0]}:
${pair[1]}`);
}
```

Fetch Post

- You can post data to a URL using

```
fetch(url, {
    method: "POST",
    headers: customHeaders,
    body: JSON.stringify(data),
})
```

- You can custom headers and you must stringify the data

How do I choose?

- Cross-browser compatibility
  - this is usually one of the jQuery key advantages
- library weight
  - if you use jQuery only for ajax, then Axios would be lighter.
- the Fetch API might be a better replacement than Axios
  - It is the new standard HTML5 API that is meant to replace XMLHttpRequest.

Questions

?

APPENDIX 1: AJAX AND JQUERY

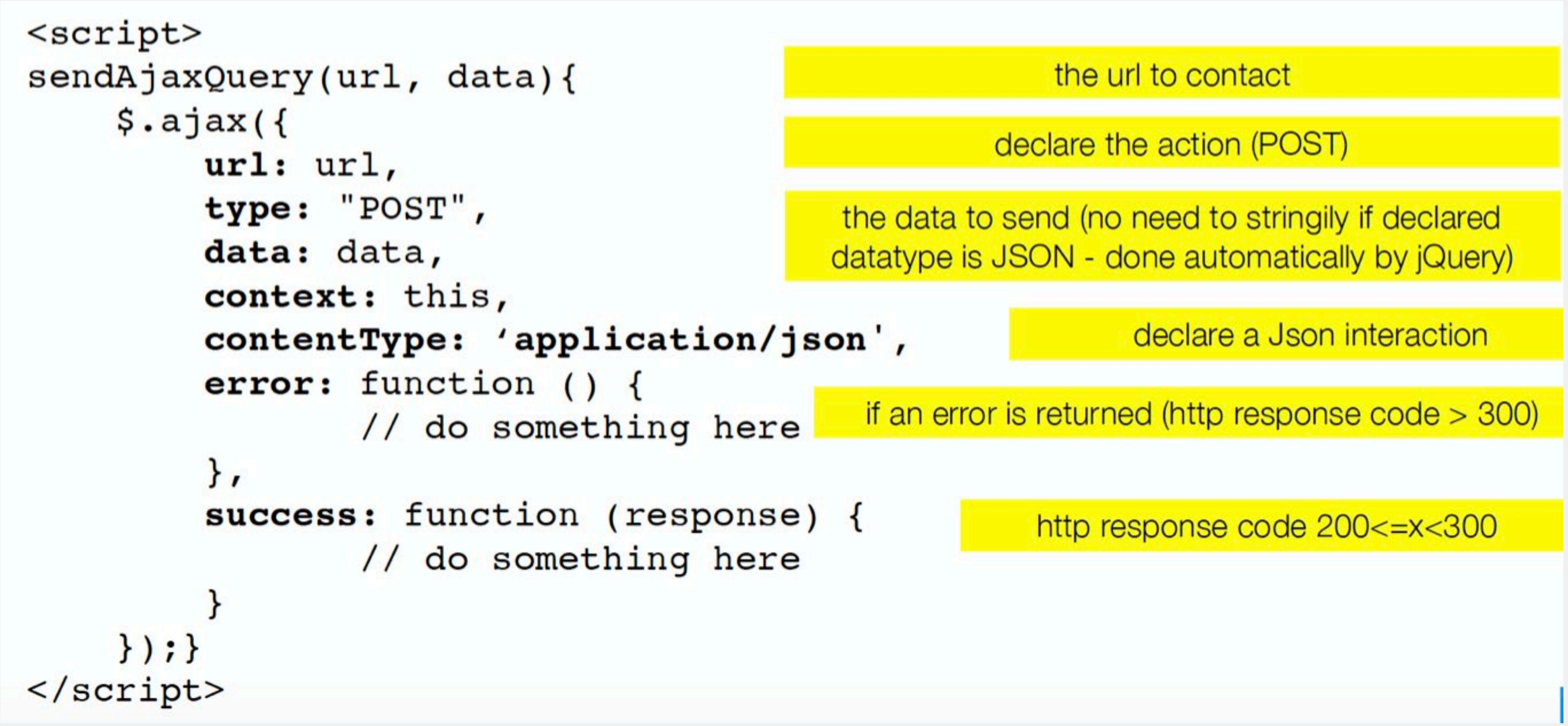
jQuery Ajax Methods: ajax()

- $.ajax() creates an ajax request
- Options
  - URL for the request
  - Async (default: true)
  - Type (GET or POST)
  - Success (Callback)
  - Error (Callback)

jQuery Ajax Methods: ajax() – cont.

- Data (to be sent to server)
- datatype (type of data you expect back)

jQuery Ajax Methods: ajax() – example



```
<script>
sendAjaxQuery(url, data){
    $.ajax({
        url: url,
        type: "POST",
        data: data,
        context: this,
        contentType: ‘application/json',
        error: function () {
               // do something here
        },
        success: function (response) {
               // do something here
        }
    });}
</script>
```

the url to contact

declare the action (POST)

the data to send (no need to stringily if declared datatype is JSON - done automatically by jQuery)

declare a Json interaction

if an error is returned (http response code > 300)

http response code 200<=x<300

jQuery Ajax Methods: load()

- loads data from a server and puts the returned data into the selected element
- Supports a callback

```
$("#demo").load(”cat.jpg");
```

jQuery Ajax Methods: load() – example HTML

```
<div id="demo">
  This is my layer
</div>


<button id="mybutton">Load New Data</button>
```

jQuery Ajax Methods: load() – example AJAX

```
<script type = "text/javascript" language = "javascript">
   $(document).ready(function() {
      $("#mybutton").click(function(event){
         $('#demo').load('data/data.txt');
      });
   });
 </script>
```

jQuery Ajax Methods: get() and post()

- $.get()
  - requests data from the server with an HTTP GET request.
- $.post()
  - requests data from the server with an HTTP POST request.