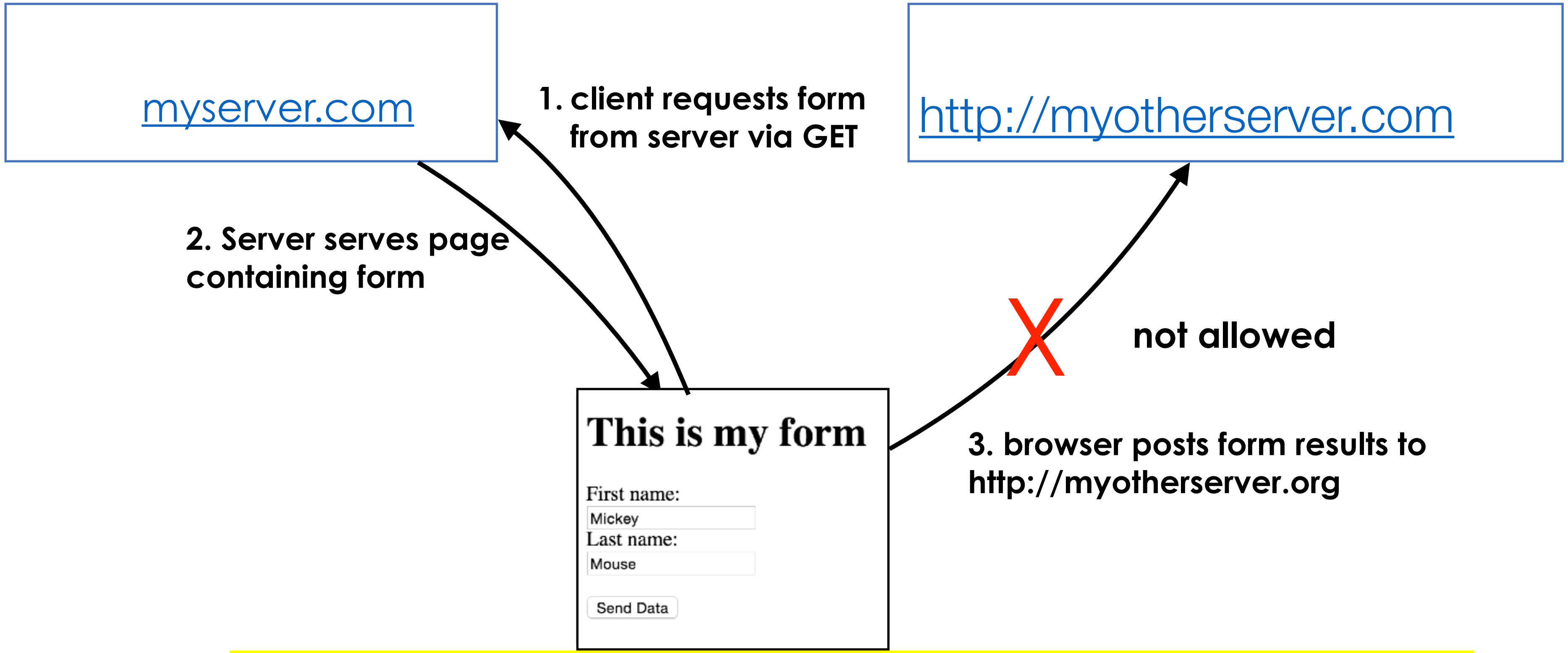
A common Issue: CORS (when the client is a browser)

Prof. Fabio Ciravegna
Dipartimento di Informatica
Università di Torino
fabio.ciravegna@unito.it

Professor Fabio Ciravegna
Department of Computer Science,
University of Sheffield
f.ciravegna@shef.ac.uk

COM3504/6504
"The Intelligent Web"

Beware! CORS



myserver.com

1. client requests form from server via GET

http://myotherserver.com

2. Server serves page containing form

not allowed

This is my form

First name: Mickey

Last name: Mouse

Send Data

3. browser posts form results to http://myotherserver.org

If the server sending the page is **http://myserver.org:63342**, you are not allowed to post to another server (e.g. http://myotherserever.org:3000).
This is to avoid man in the middle attacks. You must post to the same server.
You are not even allowed to post the the same server on another port.
Origin '**http://myserver.org:63342**' is therefore not even allowed posting to '**http://myserver.org**:3000'.

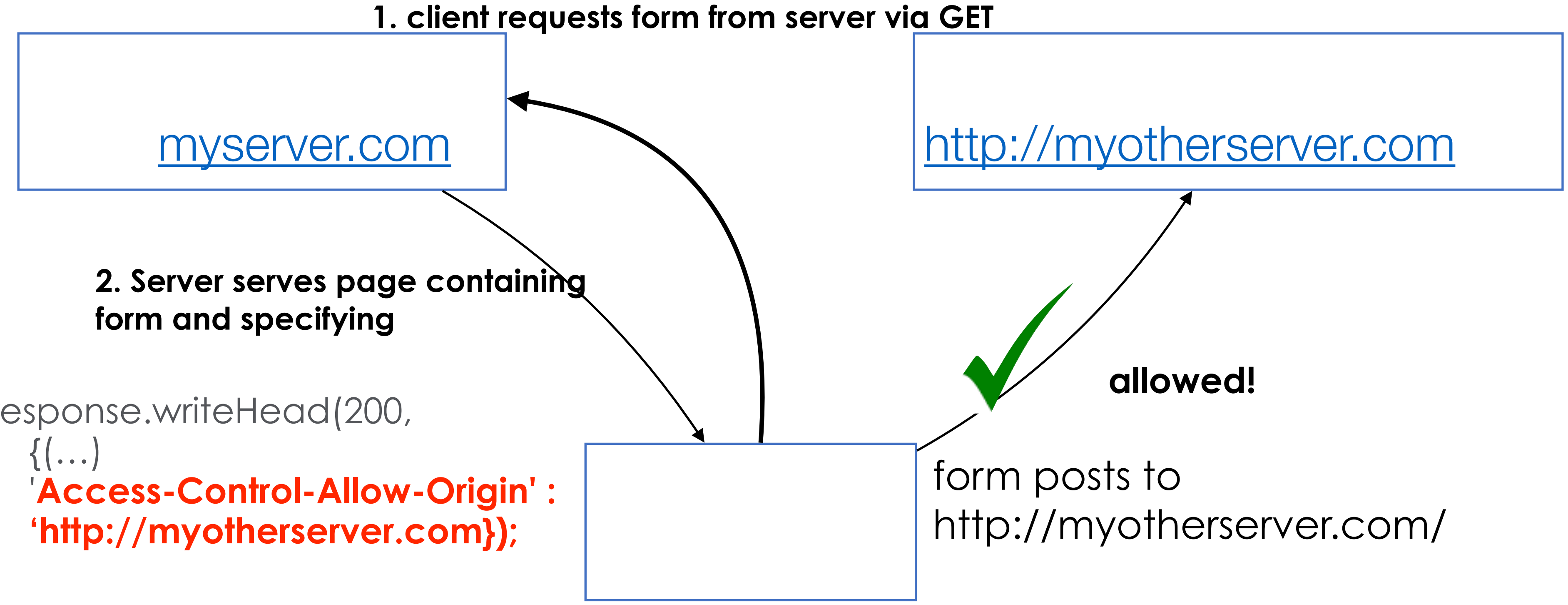
UNIVERSITÀ
DI TORINO

Beware!

- you must post to the same server serving the html file (including port!) otherwise the browser will refuse to send your request
- To avoid this block, the server *must* declare that the page is allowed to post elsewhere, i.e. the server serving the html file you must set
    - response.writeHead(200,
      {(…)
      '**Access-Control-Allow-Origin' : ‘**http://myotherserver.org:3000‘
      });

See also Cross-origin resource sharing: a simple method to perform cross-domain requests by introducing a small proxy server able to query outside the current domain
There is a simple way of doing it in node.js, php, etc.

CORS! The right way



1. client requests form from server via GET

myserver.com

http://myotherserver.com

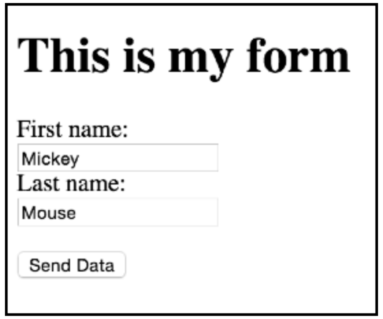
2. Server serves page containing form and specifying

```
response.writeHead(200,
    {(…)
    'Access-Control-Allow-Origin' :
    ‘http://myotherserver.com});
```

allowed!

form posts to http://myotherserver.com/

Never ever ever return 'Access-Control-Allow-Origin' : ‘*’ as it opens your server to man in the middle attacks



This is my form

First name: Mickey

Last name: Mouse

Why?

http://stackoverflow.com/questions/10636611/how-does-access-control-allow-origin-header-work

Access-Control-Allow-Origin is a CORS (Cross-Origin Resource Sharing) header.

When Site A tries to fetch content from Site B, Site B can send an Access-Control-Allow-Origin response header to tell the browser that the content of this page is accessible to certain origins. (An origin is a domain, plus a scheme and port number.) By default, Site B's pages are not accessible to any other origin; using the Access-Control-Allow-Origin header opens a door for cross-origin access by specific requesting origins.

For each resource/page that Site B wants to make accessible to Site A, Site B should serve its pages with the response header:

Access-Control-Allow-Origin: http://siteA.com
Modern browsers will not block cross-domain requests outright. If Site A requests a page from Site B, the browser will actually fetch the requested page on the network level and check if the response headers list Site A as a permitted requester domain. If Site B has not indicated that Site A is allowed to access this page, the browser will trigger the XMLHttpRequest's error event and deny the response data to the requesting JavaScript code.

Note

- Please do not confuse posting (i.e. sending data) and getting (retrieving a file)
  - of course you can have gets pointing to different servers in any html file
  - e.g. this is allowed

```
<head lang="en">
    <meta charset="UTF-8">
    <title>Ajax form</title>
    <script
  src=“http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js”>
    </script>
</head>
```

- Instead the client browser cannot **POST** data elsewhere without being allowed to do so explicitly by the server
  - so that your personal data is not sent to unauthorised people

STUDII UNIVERSITAS TAURINENSIS 1404

UNIVERSITÀ DI TORINO

Questions?

© Prof. Fabio Cravegna, Università di Torino