Cross-domain requests with CORS

Vladimir Dzhuvinov

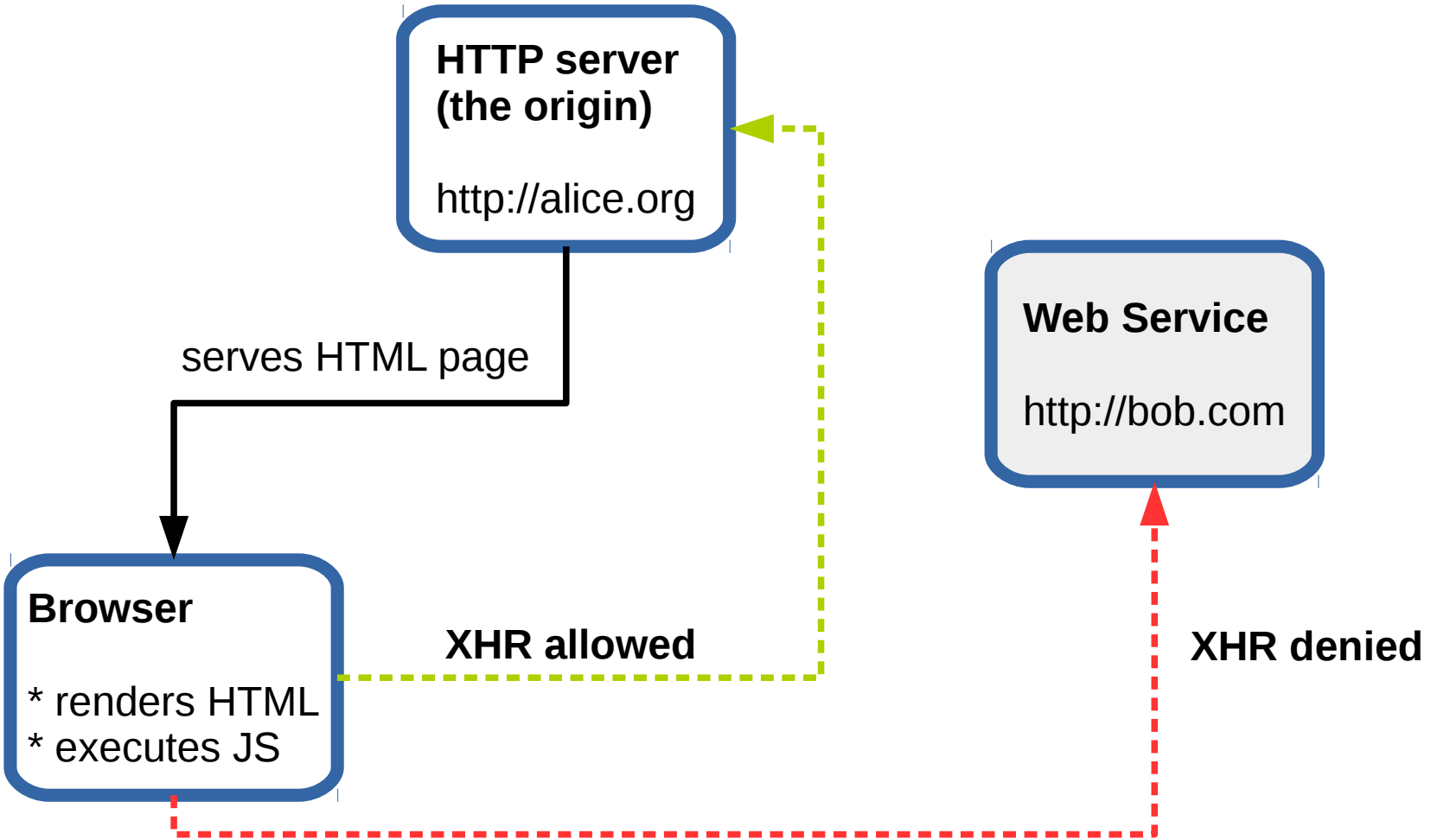
@dzhuvinov

connect2id hackafe

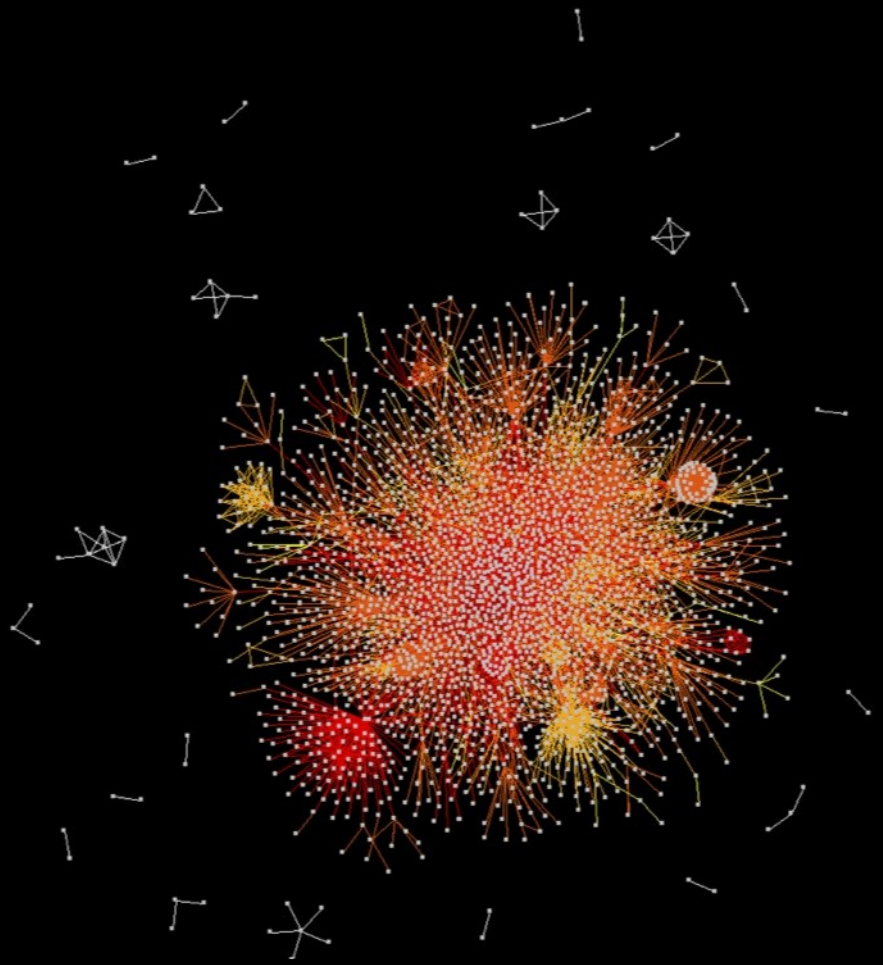
History

- **1989** The web and HTTP get invented
- **1994** Netscape Navigator
- **1995** JavaScript
- **1999** IE 5 brings **XMLHttpRequest**
- **2000+** The web becomes dynamic :-)

Dynamic web 2.0 vs same origin policy



The future of the web is cross-domain, not same origin



The first approach at solving the cross-domain problem

- JSONp
- Ugly hack
- Relies on dynamic loading of <script> tags from servers that are not on the same domain

What web gurus decided to do

- Create a new standard protocol for cross-domain XHR:
  - Define **origin**: RFC 6454
  - Define cross-domain requests: W3C Cross-Origin Resource Sharing (**CORS**)
  - Extend existing **XMLHttpRequest** object
- A 9 year effort!

Gurus such as...

The web origin concept

ORIGIN

Defined in **RFC 6454**,
published 2011,
by Adam Barth / Google

Defined by matching:

* **schema**
* **host**
* **port**

Examples of same origin:

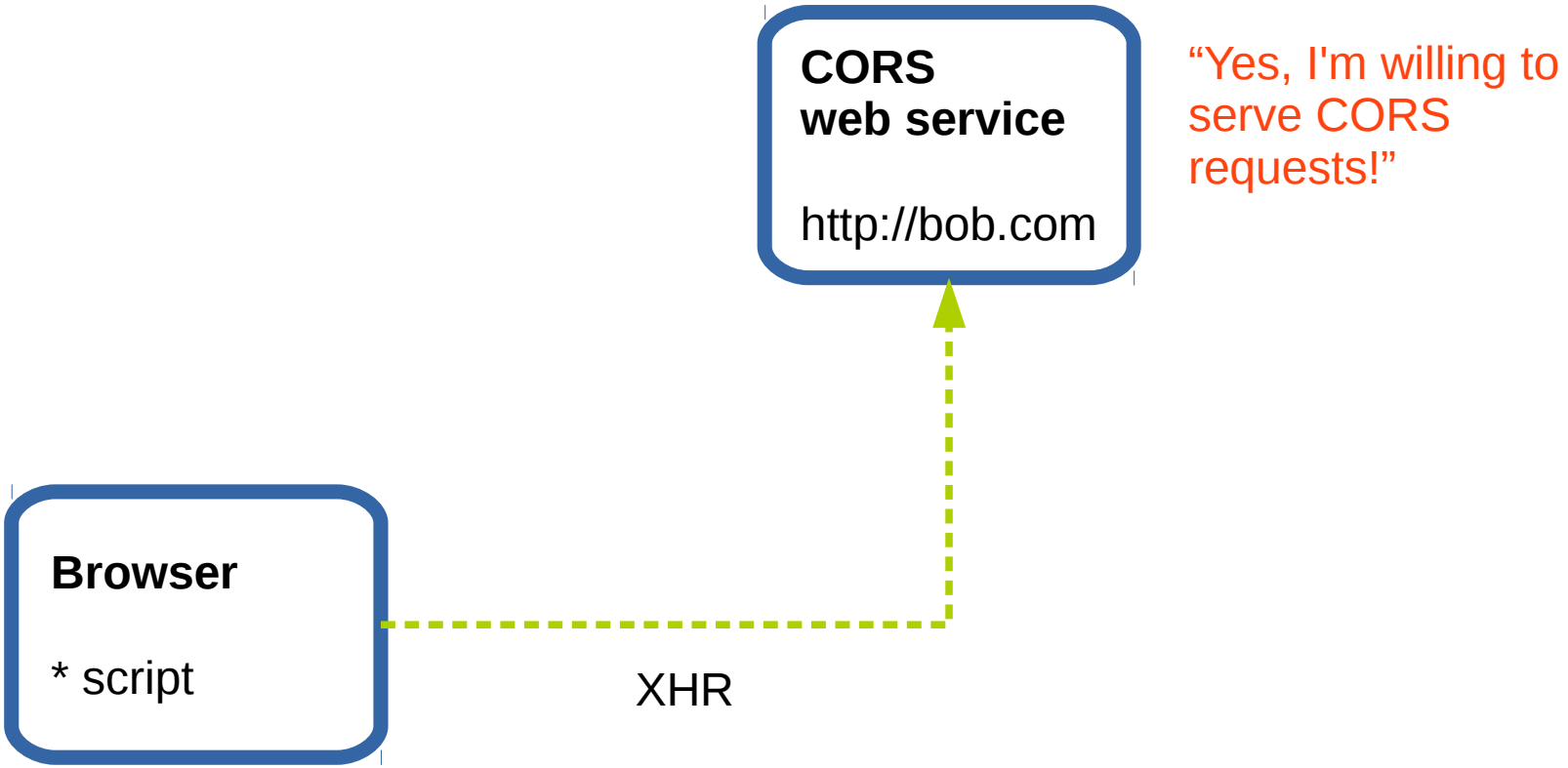
http://hackafe.org:8080/files/hello-world?q=123

http://hackafe.org:8080/files/hello-world

http://hackafe.org:8080/files/

http://hackafe.org:8080

For CORS to work HTTP servers must opt-in



Simple CORS request

- **Methods:**
  - GET
  - HEAD
  - POST
- **Request headers:**
  - Accept
  - Accept-Language
  - Content-Language
  - Content-Type:
    - text/plain
    - application/x-www-form-urlencoded
    - multipart/form-data

Simple CORS request

1. JS originating from http://alice.org:

```
var client = new XMLHttpRequest()
client.open("GET", "http://bob.com/hello")
client.onreadystatechange = function() { /* do something */ }
client.send()
```

2. HTTP Request browser → CORS server:

```
GET /hello HTTP/1.1
Host: http://bob.com
Origin: http://alice.org
```

3. HTTP Response CORS server → browser:

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: http://alice.org
Content-Type: text/plain

Hello world!
```

Preflight request

- For methods other than GET, HEAD and POST
- For credentials, such as cookies, HTTP basic and tokens
- For request headers such as Content-Type: application/json
- To expose non-simple response headers to the JavaScript, e.g. X-Custom-Header

Preflight request with HTTP OPTIONS

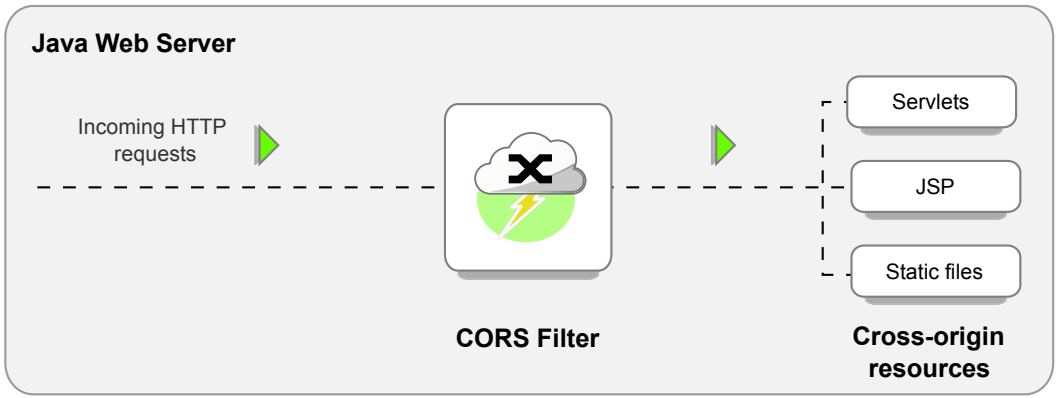
1. HTTP Request browser → CORS server:

OPTIONS /hello HTTP/1.1
Host: http://bob.com
Origin: http://alice.org
Access-Control-Request-Method: PUT
Access-Control-Request-Headers: Content-Type, Authorization

2. HTTP Response CORS server → browser:

HTTP/1.1 200 OK
Access-Control-Allow-Origin: http://alice.org
Access-Control-Allow-Methods: GET, POST, PUT, DELETE
Access-Control-Allow-Headers: Content-Type, Authorization
Access-Control-Expose-Headers: X-Custom-Header
Access-Control-Allow-Credentials: true
Access-Control-Max-Age: 3600

Handling CORS on the server side



http://software.dzhuvinov.com/cors-filter.html

You don't need to code anything, use existing CORS filters or modules

How to detect CORS support in the browser

```
function browserSupportsCors() {

        if ("withCredentials" in new XMLHttpRequest())
                return true;

        else if (typeof XDomainRequest == "object")
                return true;

        else
                return false;
}
```

CORS support

Browsers supporting CORS

All major browsers support CORS. The reported penetration among users is at 89% as of November 2013.


Firefox 3.5+


Internet Explorer 8+ †


Google Chrome 3+


Apple Safari 4+


Opera 12+

† Partial support via the XDomainRequest object. Version 10 of IE is expected to have full CORS support integrated into the common XMLHttpRequest object.

Share cookies

```
var xhr = new XMLHttpRequest();

var url = 'http://bar.other/resources/credentialed-content/';

xhr.open('GET', url, true);

xhr.withCredentials = true;

xhr.onreadystatechange = handler;

xhr.send();
```

Thank you!

Q + A?