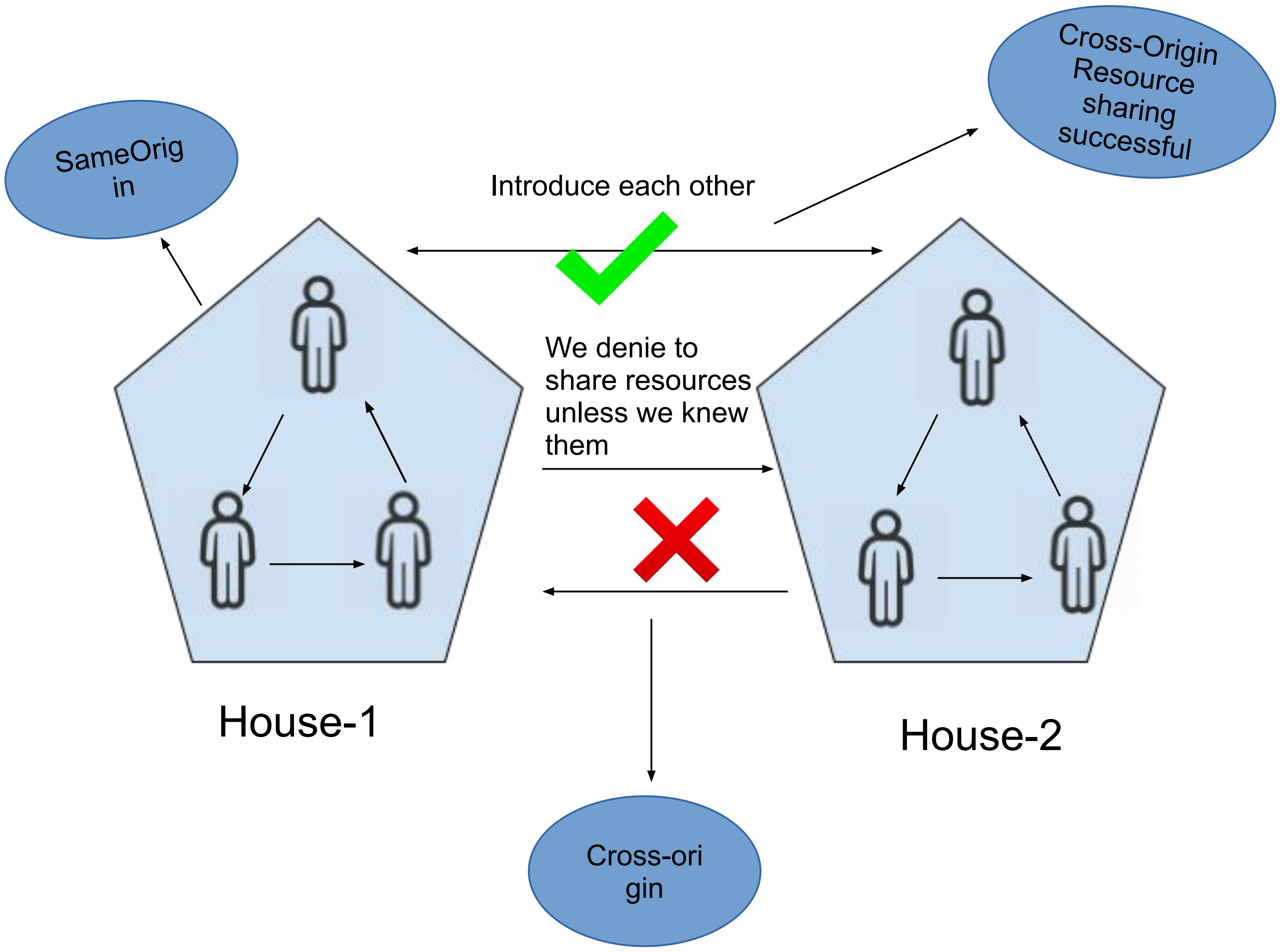
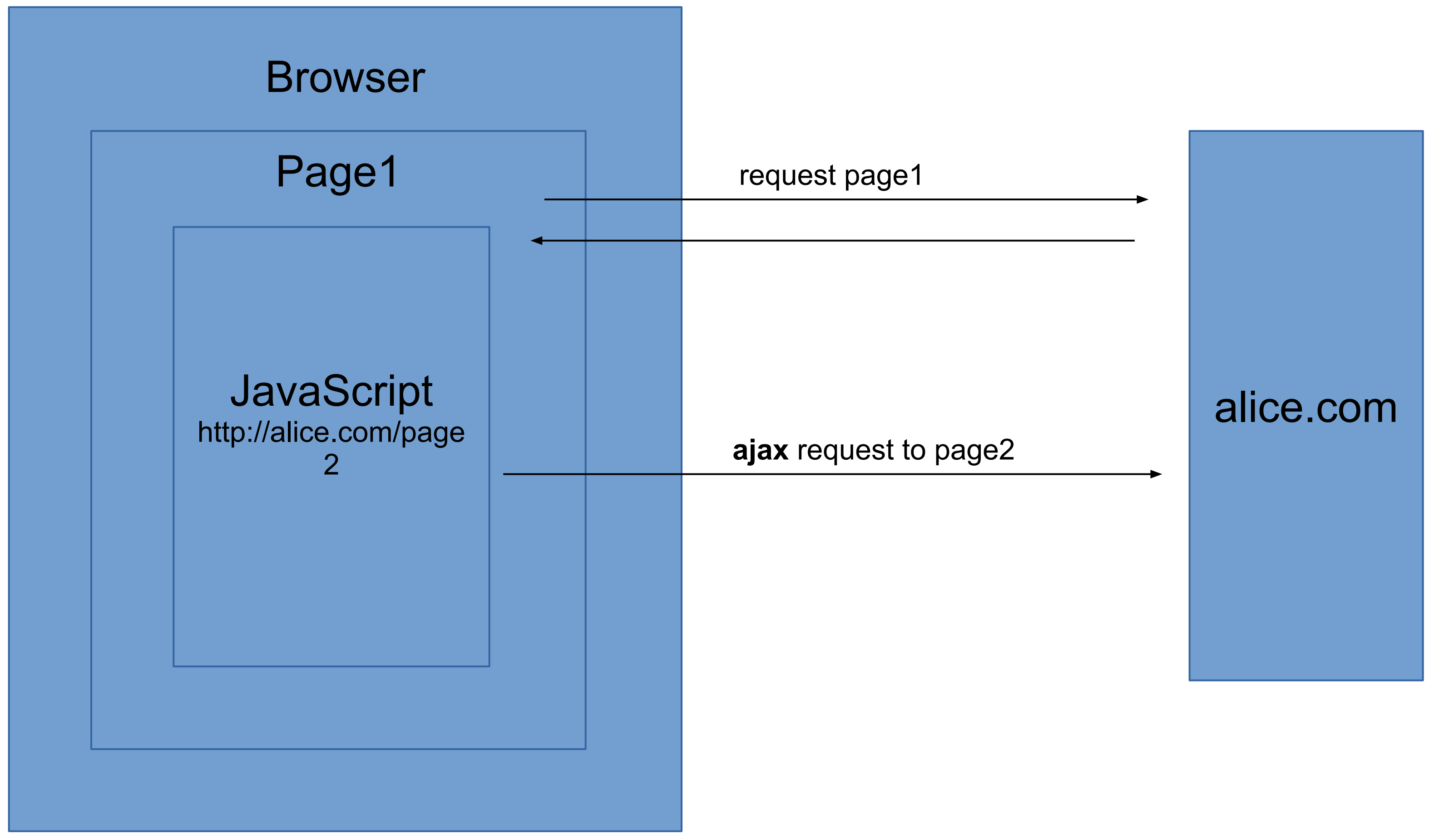
Technical insight into concepts and terminologies behind cors

T.Srujan
Development Engineer

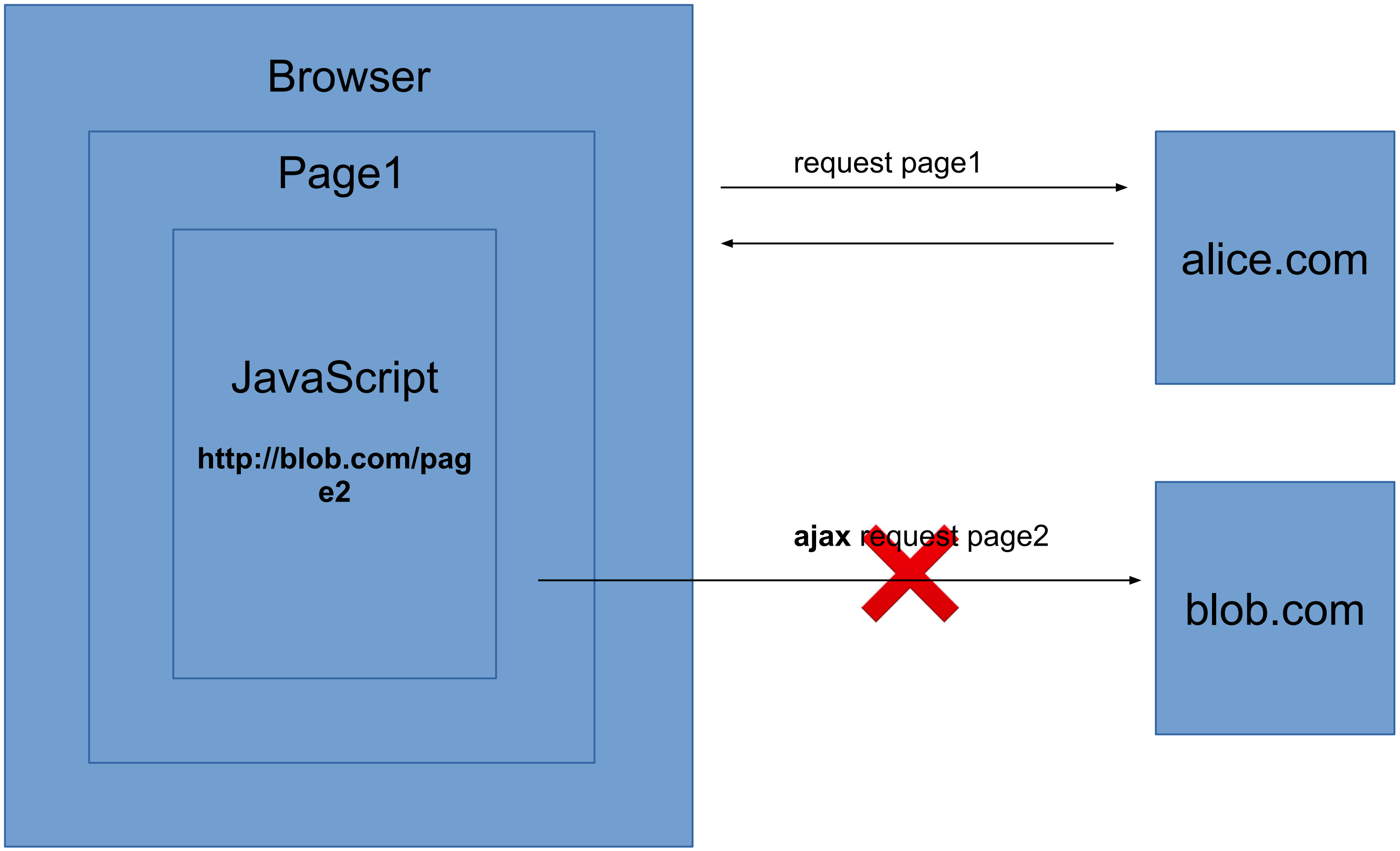
Resource Sharing


SameOrig
in
Introduce each other
Cross-Origin Resource sharing successful
We denie to share resources unless we knew them
House-1
House-2
Cross-ori
gin

Same-Origin Resource Sharing



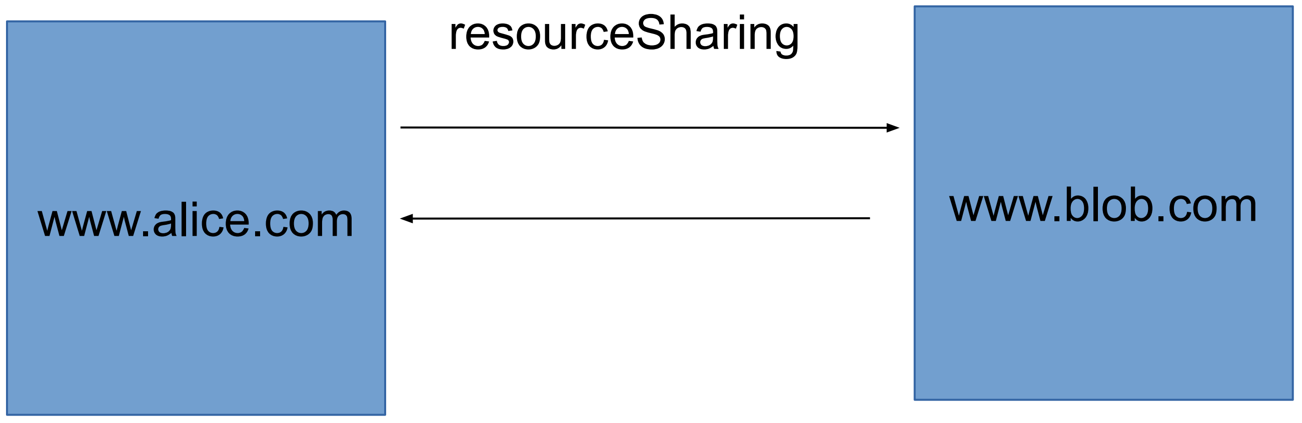
Cross-Origin Resource Sharing



cors

CORS

Cross-Origin Resource Sharing is W3C spec that allows cross-domain communication from the browser.



www.blob.com

Request Headers Used in CORS Flow

- Origin
- Access-Control-Request-Method
- Access-Control-Request-Headers

Response Headers Used in CORS Flow

- Access-Control-Allow-Origin
- Access-Control-Allow-Methods
- Access-Control-Allow-Headers
- Access-Control-Allow-Credentials
- Access-Control-Exposed-Headers
- Access-Control-Max-Age

Simple cors requests

Simple CORS Request

- Simple CORS Request

  HTTP Method

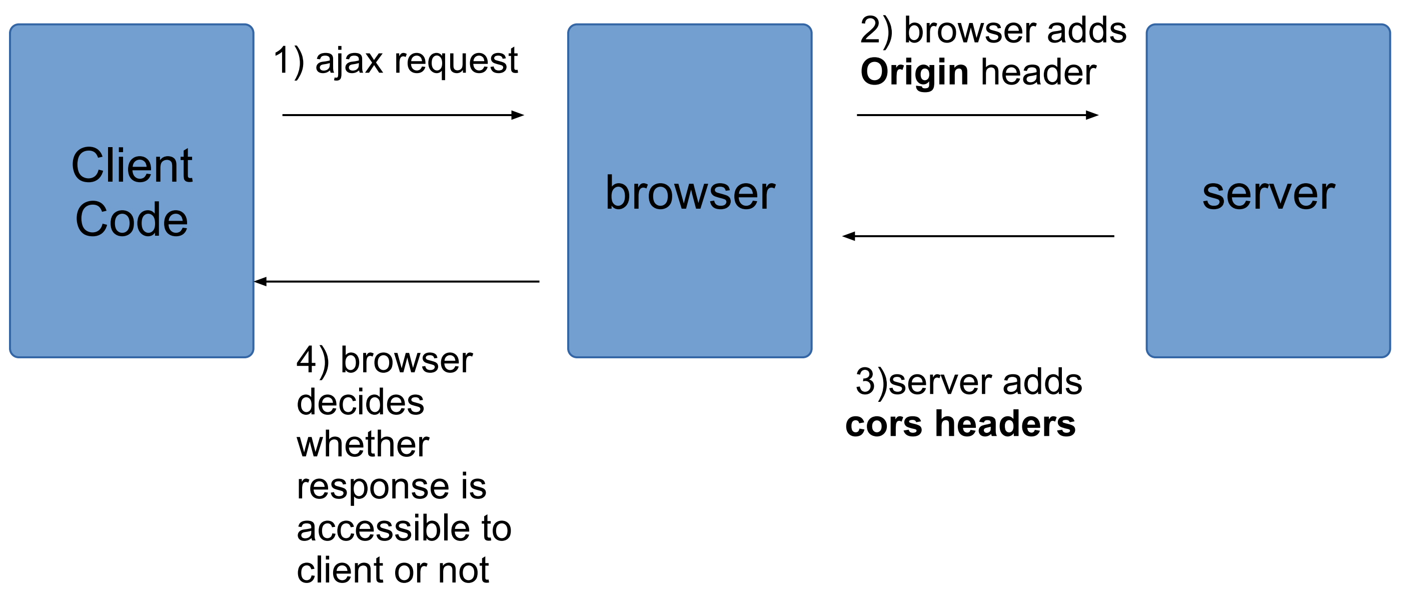
  - GET, POST, HEAD

  Content-Type Header

  - application/x-www-form-urlencoded
  - multipart/form-data
  - text/plain

  No Custom Request Headers

CORS LifeCycle


Client Code
1) ajax request
browser
2) browser adds **Origin** header
server
3)server adds **cors headers**
4) browser decides whether response is accessible to client or not

Origin

- Origin request header is central to CORS. Client identifies itself to server by using the origin header.
- A CORS request must have origin header.
- Origin = scheme + host + port
- Origin = null, when origin can't be determined.

  For example when a file is opened from your file system browser sets the origin to null.

Responding to CORS Request

- Access-Control-Allow-Origin

Server uses this header to approve the request. This header

must be present on every successful CORS response. This value can be either a wild card or origin value as shown below.

```
Access-Control-Allow-Origin: *

Access-Control-Allow-Origin: http://alice.com
```

Not-So-Simple CORS Requests

- Not-So-Simple Requests
  - HTTP Method
    - PUT, DELETE
    - Content-Type with headers other than
      - application/x-www-form-urlencoded
      - text/plain
      - multipart/form-data
  - Contains custom request headers.

- These requests needs to ask for servers permissions before making the actual request. Browser asks for permissions by using **preflight request**.

preflight requests

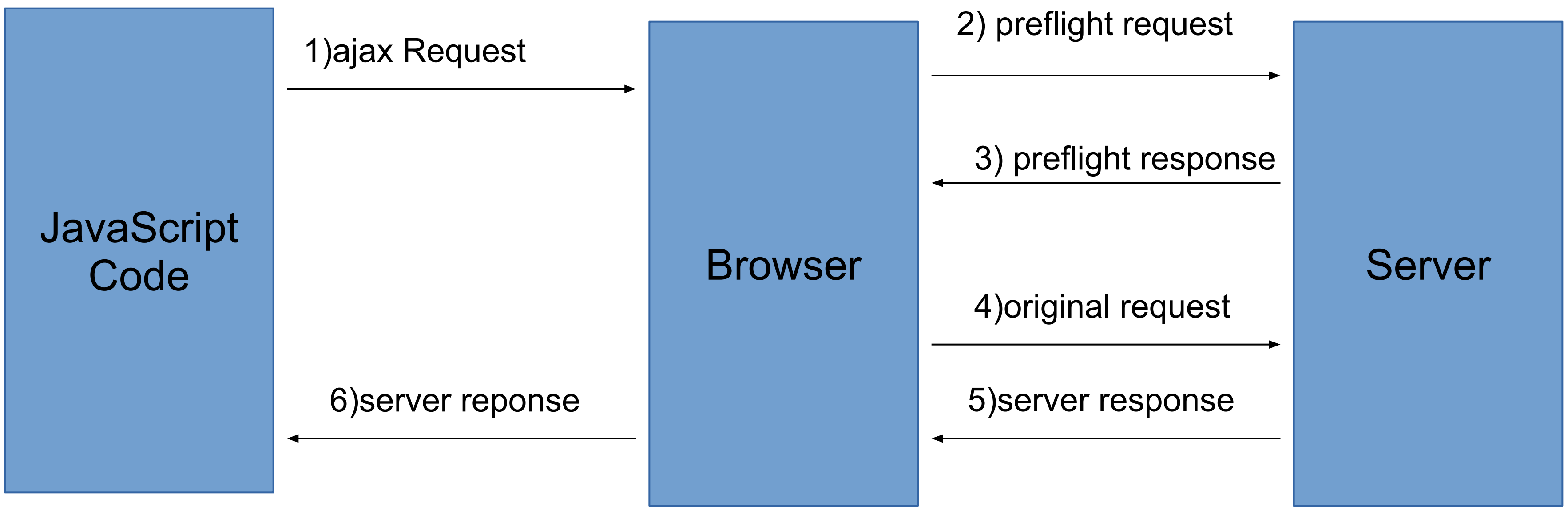
Preflight Request

Preflight Request

- A preflight request asks for server permission before making the original request.
- It contains metadata about the request, such as which HTTP method is used and if any custom request headers are sent.

Preflight Request

- Ask for permission before making the actual request.



Preflight Request

How to distinguish preflight and normal request?

Preflight Request Characteristics

- HTTP **OPTIONS** method
- **Origin** request header
- **Access-Control-Request-Method**
- **Access-Control-Request-Headers** is optional. Used when custom headers are sent in the request.

Preflight Request

How does server grant the permission ?

Access-Control-Allow-Methods

Response header with the list of methods that server allows.

Eg: Access-Control-Allow-Methods: PUT, DELETE

Access-Control-Allow-Headers

Specifies list of headers allowed in the request.

Eg: Access-Control-Allow-Headers: header1, header2

Preflight Scenarios

Request
Origin :http://localhost:8001 Access-Control-Request-Method : PUT Access-Control-Request-Header : userstatus

Response	Reason
HTTP 200 OK	No Access-Control-Allow-Origin header
HTTP 200 OK Access-Control-Allow-Origin : *	No Access-Control-Allow-Methods header
HTTP 200 OK Access-Control-Allow-Origin: * Access-Control-Allow-Method: DELETE	No Access-Control-Allow-Headers method
HTTP 200 OK Access-Control-Allow-Origin: * Access-Control-Allow-Method: PUT Access-Control-Allow-Header: userstatus	Everything matches. Request is accepted.

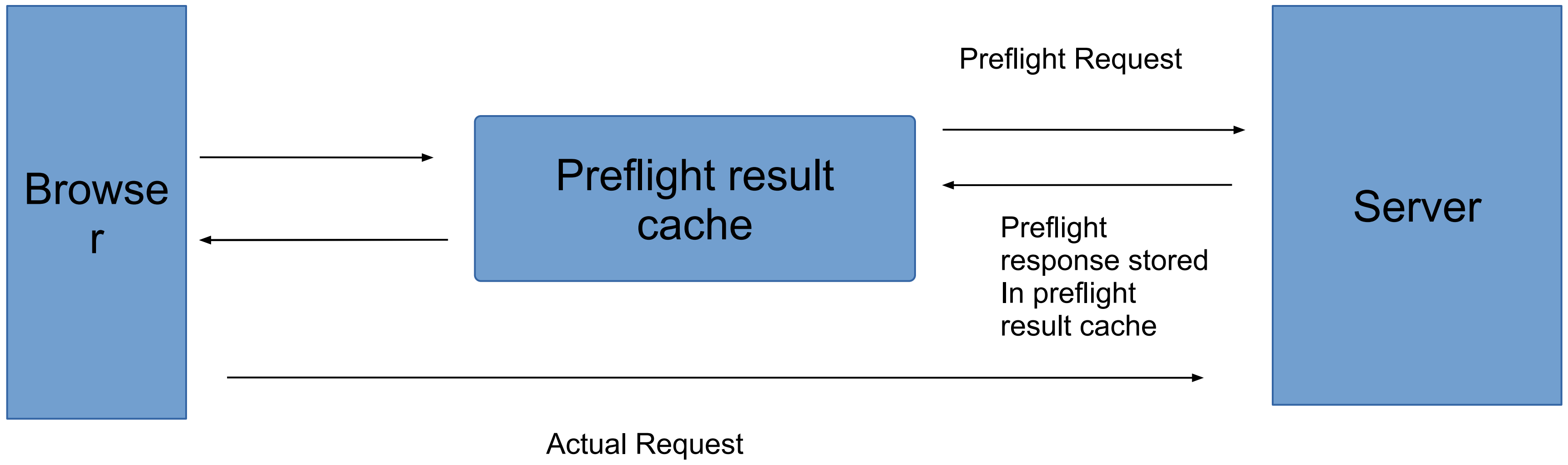
Preflight Result Cache

- **Dis-Advantage of Preflight Request**
  - Is a performance concern as two Http requests are invoked, one for the preflight and the second for actual request.

- **Preflight Cache**
  - To reduce the number of preflight requests, preflight responses are caches in preflight result cache.
  - It maintains a map of preflight results.
    - URL + origin → preflightResult

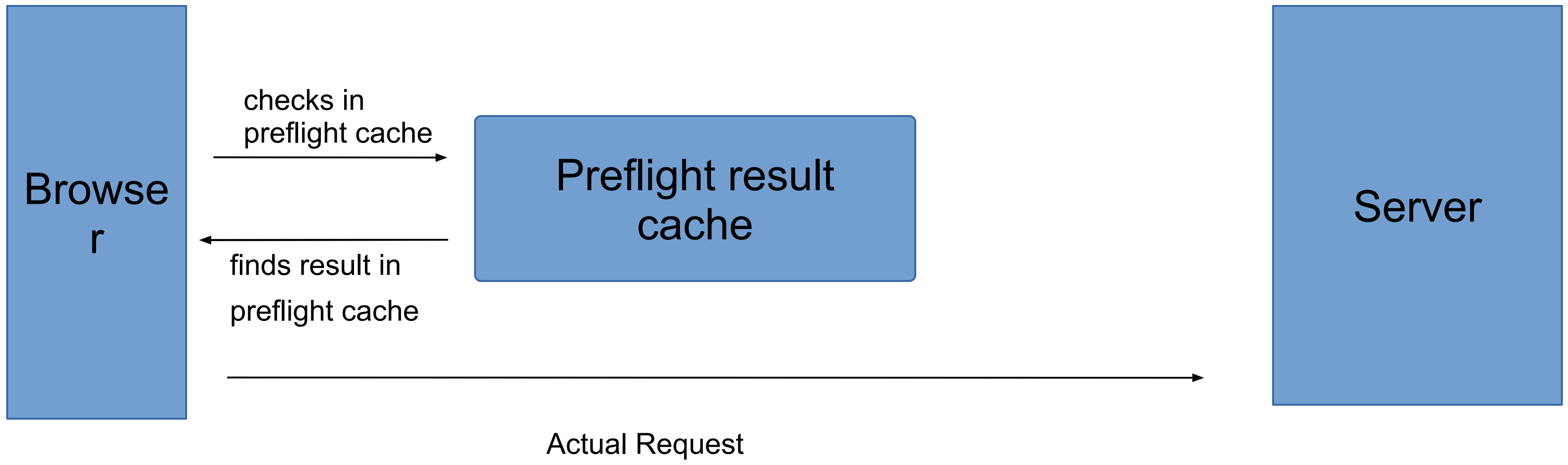
Preflight Result Cache

FirstRequest



Preflight Result Cache

Same Request Second time



Preflight Result Cache Expire

- Access-Control-Max-Age
  - This response header indicates how long, in seconds a response can be cached.
  - Response gets cached only if browser accepts the response.

- Cache Storage time in different browsers
  - Firefox → not more than 24 hours.
  - chrome, opera, safari → max of 5 mins.

User credentials with cors

User Credentials

- By default, CORS doesn’t add user credentials such as cookies on the request.
- Client must set the **withCredentials** property to true, to indicate that cookies are sent in the request.
- Server indicates that it can receive user credentials in the request by setting **Access-Control-Allow-Credentials** header to true.

User Credentials

withCredentials	Acces-Controll-Allow-Credentials	UserCredentials from Client	Server support for UserCredentials
false	false	Cookies aren't included in the request. [Allow]	Server doesn't allow cookies.
true	true	Cookies are included in the request. [Allow]	Server allows cookies.
false	true	Cookies aren't included in request. [Allow]	Server allows cookies.
true	false	Cookies are included in request.[Reject]	Server doesn't allow cookies.

User Credentials

- If **Access-Control-Allow-Credentials** is set to true,

server must provide actual origin value in **Access-Control-Allow-Origin.**

Access-Control-Allow-Credentials	Access-Control-Allow-Origin	Request status
true	*	**Reject**
true	http://abc.com	**Accept**

Reading Response Headers

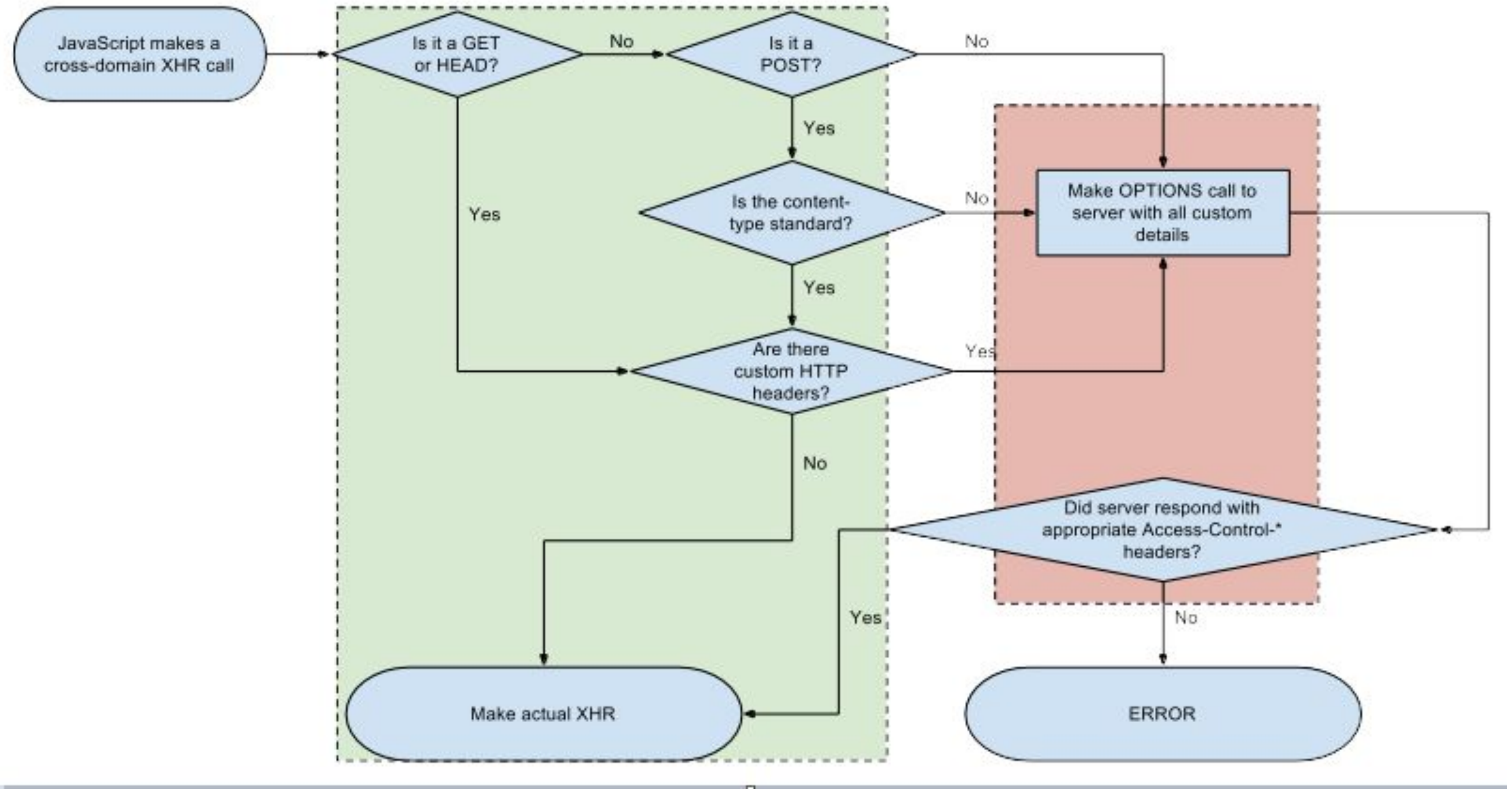
- Cross-Origin requests have limitations on which response headers can be read by client.
- By default, only simple response headers can be accessed by the client.
- Simple Response headers are:
  - Cache-Control, Expires, Content-Language, Content-Type, Last-Modified, Pragma.

Reading Response Headers

- If you want clients to access headers apart from simple headers, server must list them using **Access-Control-Exposed-Headers**.
  - Access-Control-Expose-Headers: <header-name>, <header-name>

- This header ensures that client can only read response headers intended by the server.

CORS Flowchart


JavaScript makes a cross-domain XHR call
Is it a GET or HEAD?
No
Is it a POST?
No
Yes
Yes
Is the content-type standard?
No
Make OPTIONS call to server with all custom details
Yes
Are there custom HTTP headers?
Yes
No
Did server respond with appropriate Access-Control-* headers?
Yes
No
Make actual XHR
ERROR

Benefits of CORS

- It opens up access of our API to wider audience.
- Servers stay in charge of who can access the API’s.
- Flexibility
  - Which **domains** are allowed to make requests.
  - Which **methods** are supported.
  - Which **headers** are allowed.
  - Whether request may include **cookies** are not.
- Makes it easy for client developers to use.

Thank You!