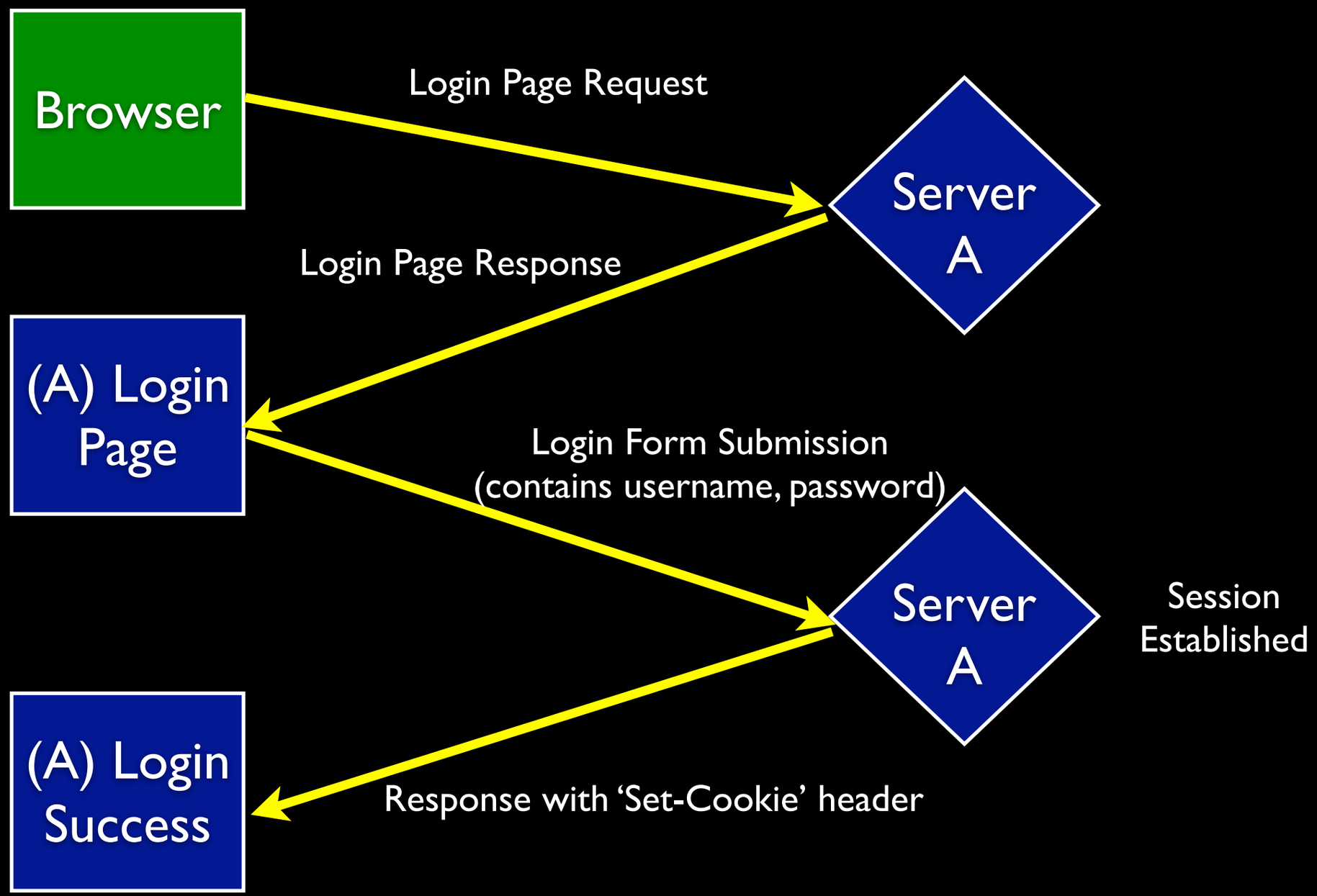
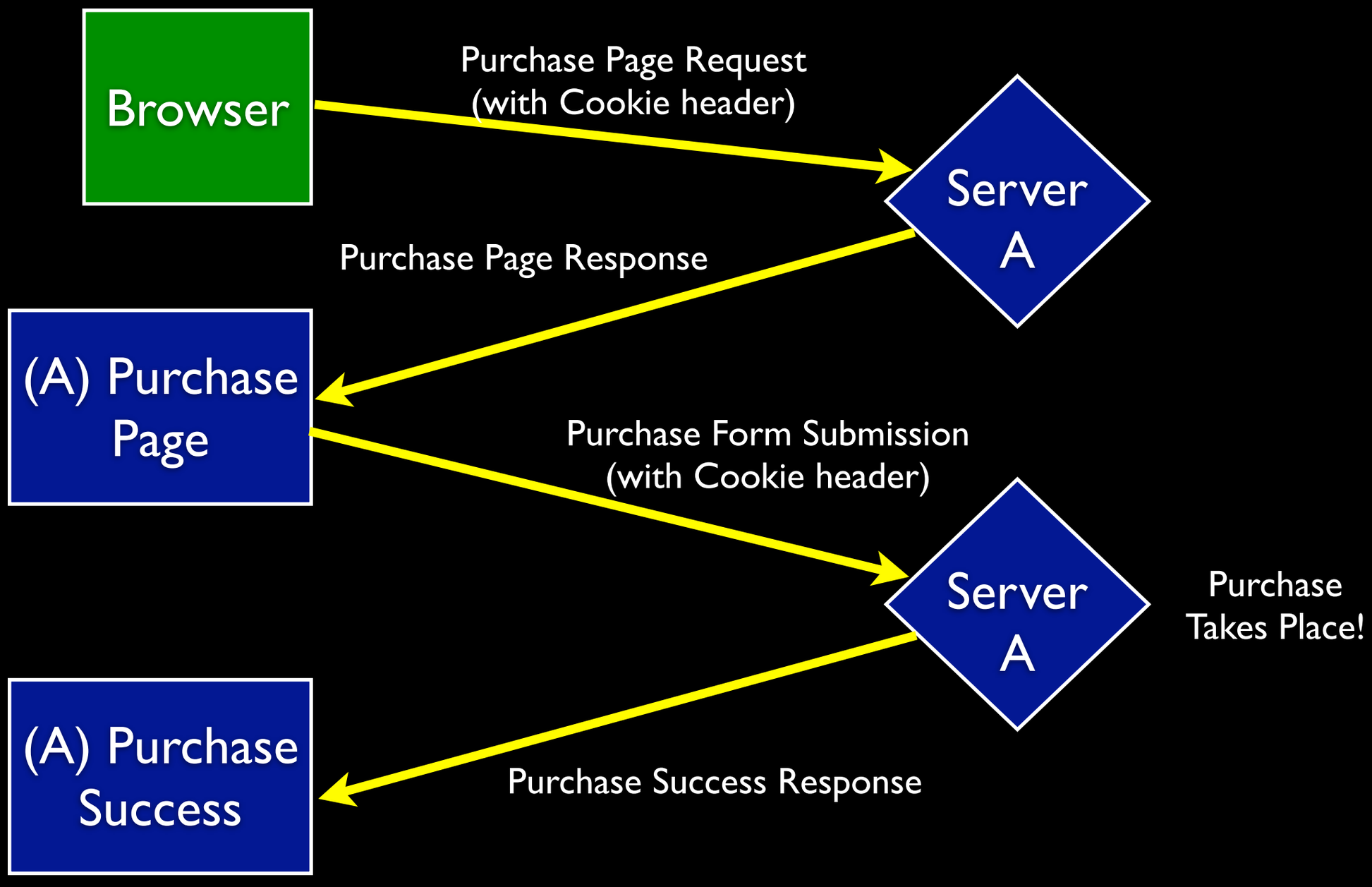
CORS Background

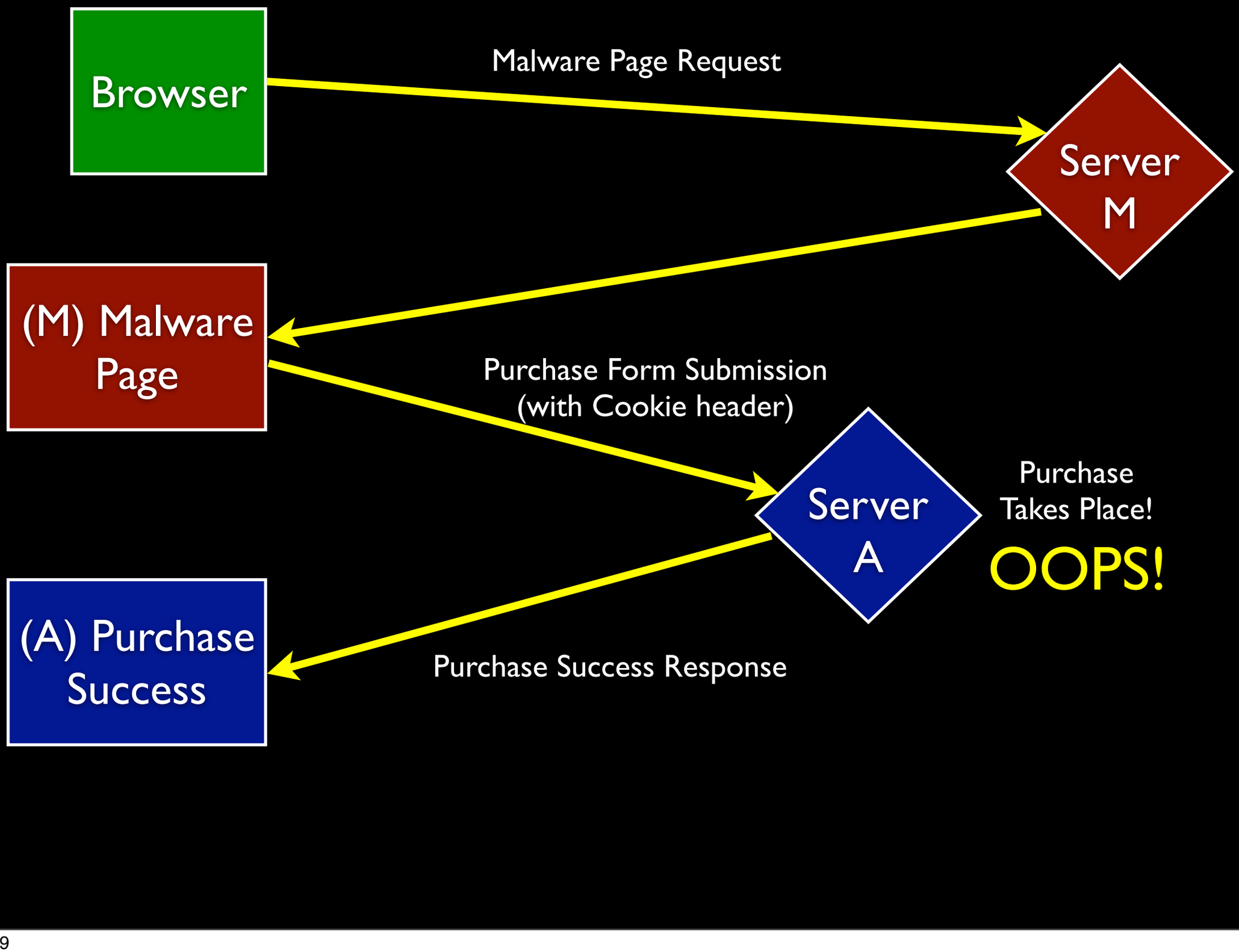
Cookies: Login



Cookies: Buying Stuff



Cookies: CSRF



Can’t Easily Avoid

Can't Easily Avoid

- Need second factor to tell if submission actually came from Site A

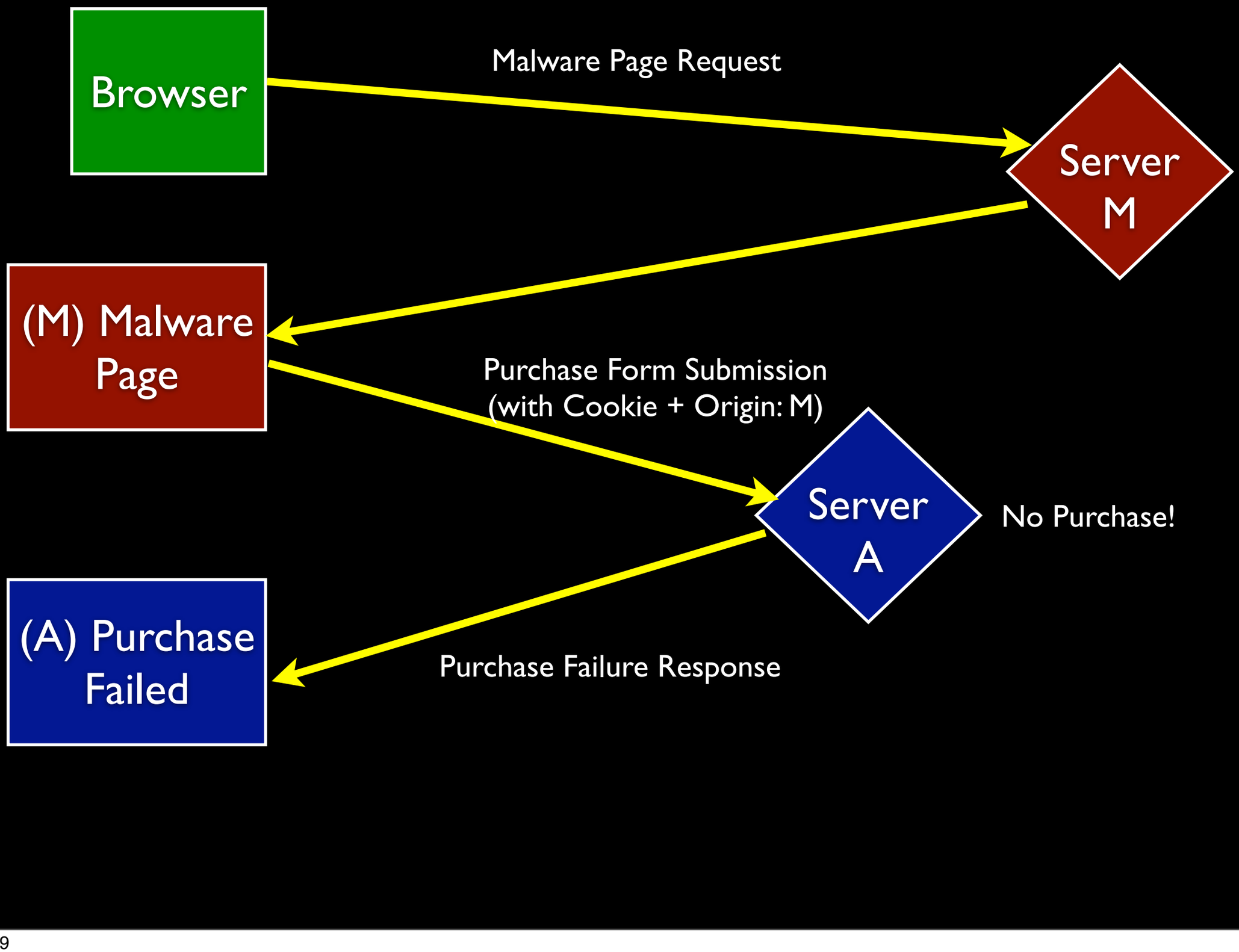
Can't Easily Avoid

- Need second factor to tell if submission actually came from Site A
- Origin

Can't Easily Avoid

- Need second factor to tell if submission actually came from Site A
  - Origin
  - Secret token (embedded in form)

CSRF: Origin Defense



CORS Scenario

- I want to let Site A (an upcoming events side) add calendar events to Site B (my calendar)

Requirements

Requirements

- Grant permission just once

Requirements

- Grant permission just once
- No manual steps to copy data between sites

Requirements

- Grant permission just once
- No manual steps to copy data between sites
- “AJAX” UI (avoid full page loads)

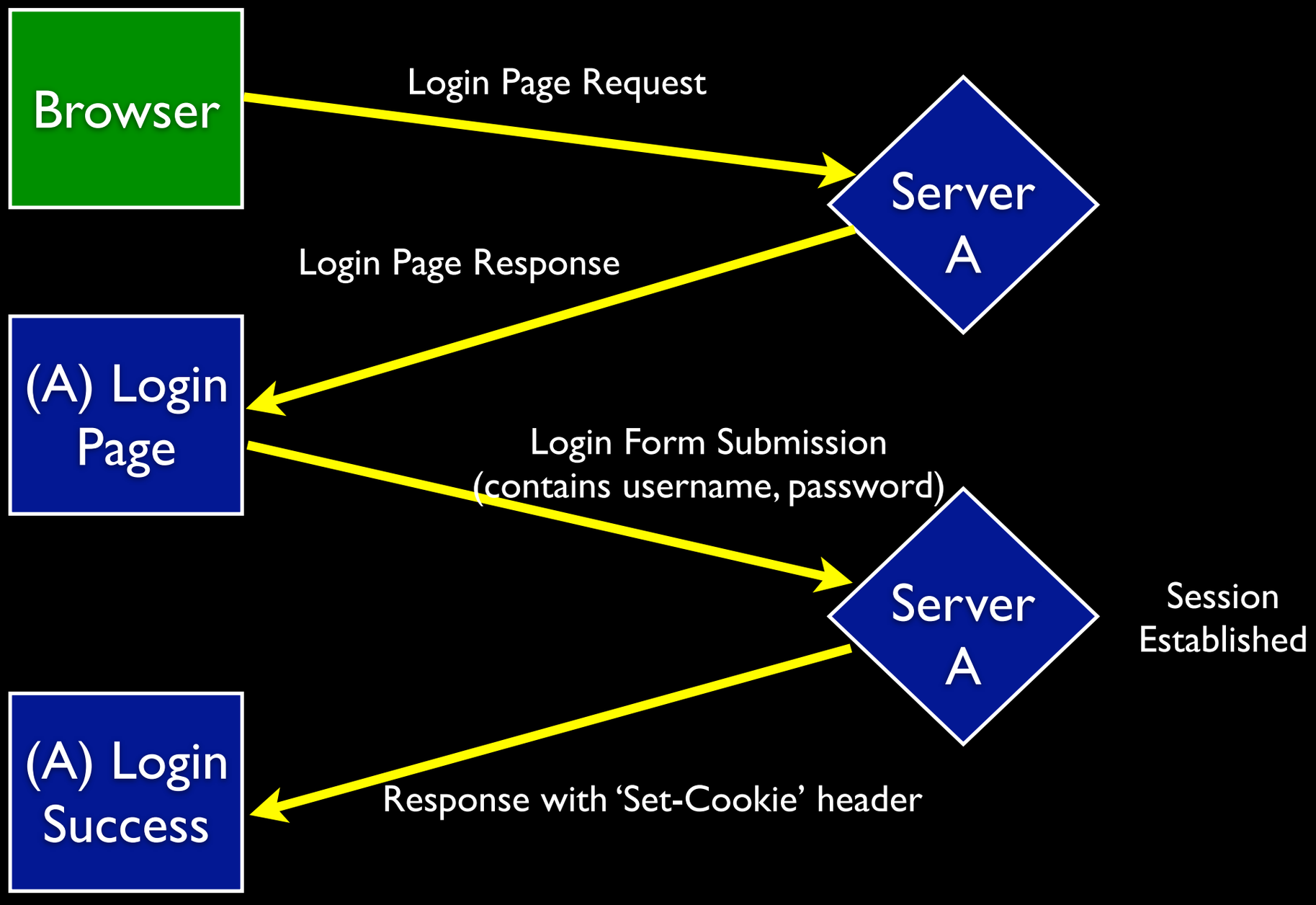
Requirements

- Grant permission just once
- No manual steps to copy data between sites
- “AJAX” UI (avoid full page loads)
- No server-to-server communication

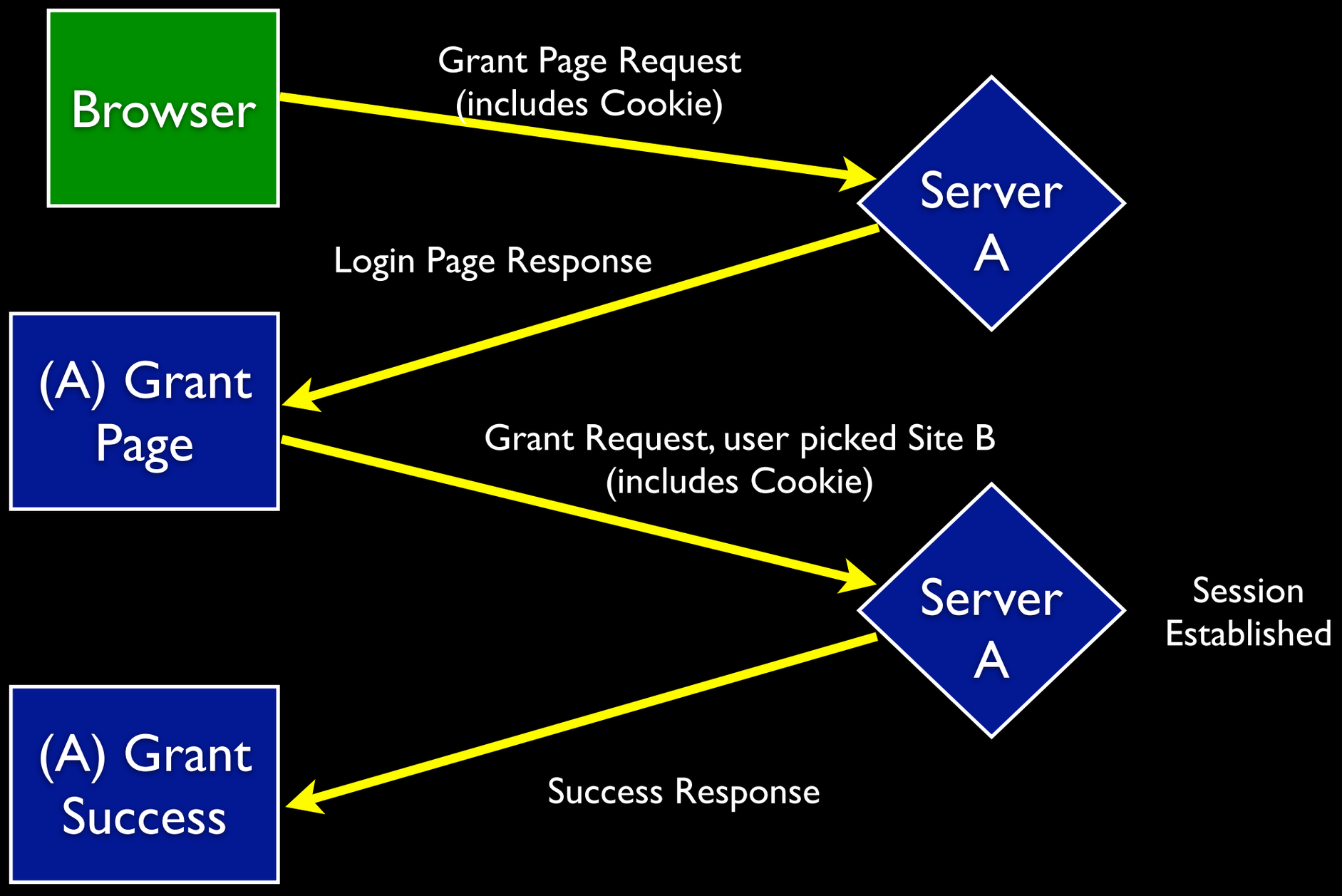
Requirements

- Grant permission just once
- No manual steps to copy data between sites
- “AJAX” UI (avoid full page loads)
- No server-to-server communication
- No need for prior bilateral arrangement between A and B, just published API

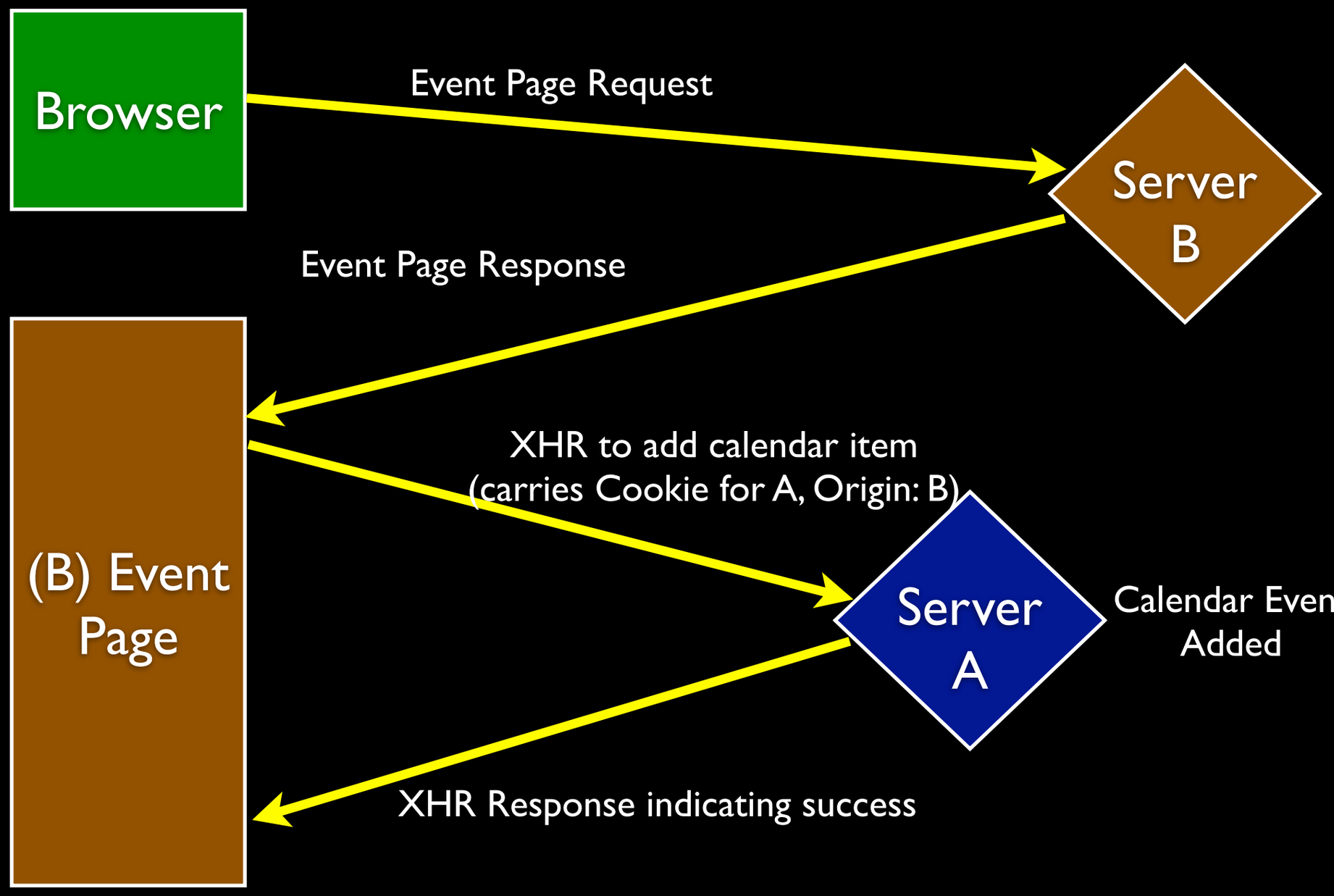
CORS: Login


Browser
Login Page Request
Server A
Login Page Response
(A) Login Page
Login Form Submission
(contains username, password)
Server A
Session Established
(A) Login Success
Response with 'Set-Cookie' header

CORS: Access Grant



CORS: Request



No Confused Deputy

No Confused Deputy

- Server M can't forge Origin in the browser

No Confused Deputy

- Server M can't forge Origin in the browser
- Server M can't send session cookie for A outside the browser

No Confused Deputy

- Server M can't forge Origin in the browser
- Server M can't send session cookie for A outside the browser
- Combination of Origin and Cookie soundly identifies combination of user and site

Fancier Scenarios Can Have CD

Fancier Scenarios Can Have CD

- Site A asking Site B to do something on Site C

Fancier Scenarios Can Have CD

- Site A asking Site B to do something on Site C
- Can also have Confused Deputy without CORS - for example poorly implemented secret tokens

How to Avoid Confused Deputy

How to Avoid Confused Deputy

- Don’t be a deputy!

How to Avoid Confused Deputy

- Don't be a deputy!
- Never ask one server to do something on behalf of another.

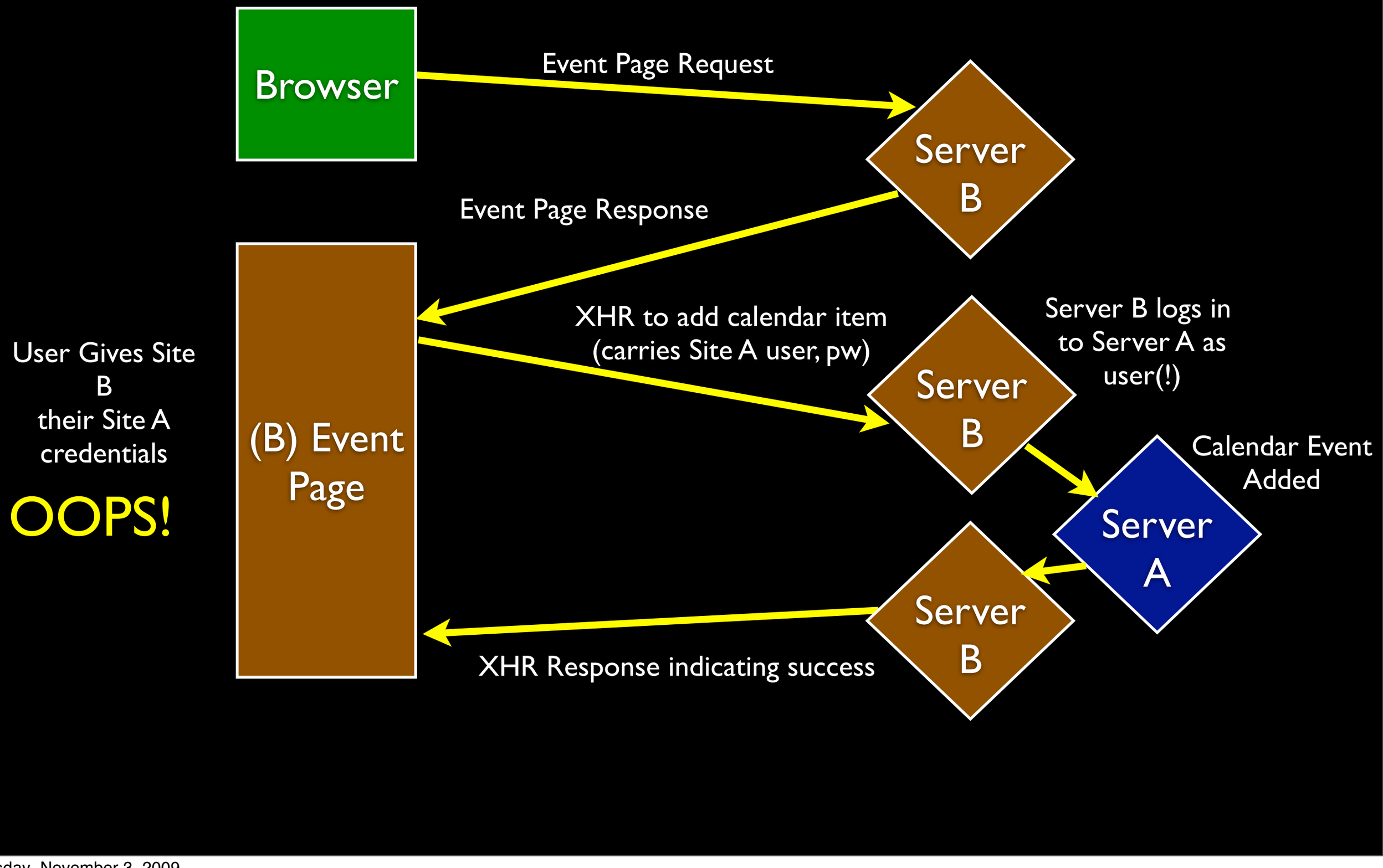
How to Avoid Confused Deputy

- Don't be a deputy!
- Never ask one server to do something on behalf of another.
- If you must...

How to Avoid Confused Deputy

- Don't be a deputy!
- Never ask one server to do something on behalf of another.
- If you must...
  - Guarantee that requests on behalf of a third party look different from your own

The Bad Way


Browser
Event Page Request
Server B
Event Page Response
(B) Event Page
User Gives Site B their Site A credentials
OOPS!
XHR to add calendar item (carries Site A user, pw)
Server B
Server B logs in to Server A as user(!)
Server A
Calendar Event Added
Server B
XHR Response indicating success

Non-CORS Solutions

Non-CORS Solutions

- For example, OAuth

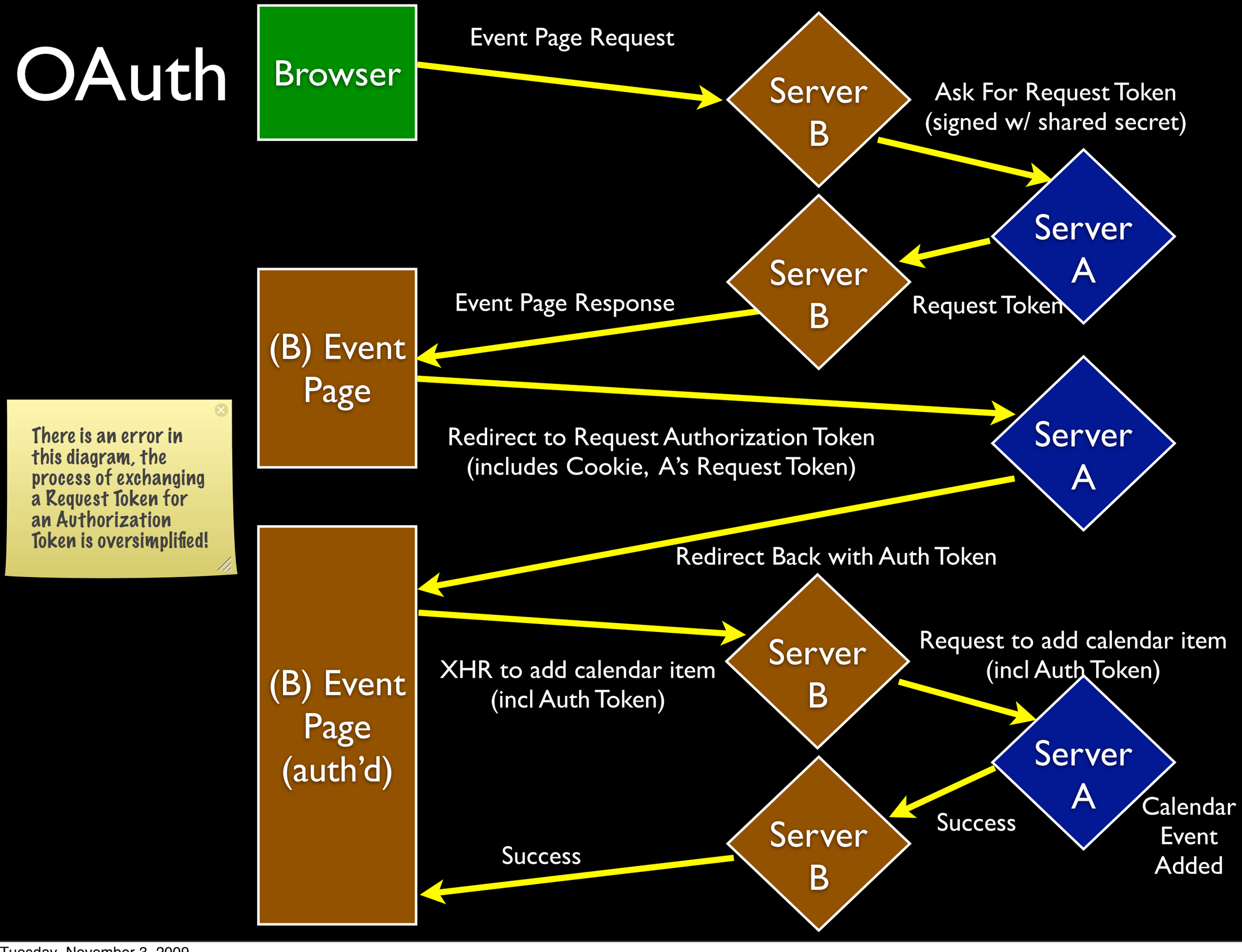
Non-CORS Solutions

- For example, OAuth
- Generally require server-to-server communication

Non-CORS Solutions

- For example, OAuth
- Generally require server-to-server communication
- Relies on bilateral agreement (shared secret)

OAuth


Browser
Event Page Request
Server B
Ask For Request Token (signed w/ shared secret)
Server A
Server B
Request Token
Event Page Response
(B) Event Page
There is an error in this diagram, the process of exchanging a Request Token for an Authorization Token is oversimplified!
Redirect to Request Authorization Token (includes Cookie, A's Request Token)
Server A
Redirect Back with Auth Token
(B) Event Page (auth'd)
XHR to add calendar item (incl Auth Token)
Server B
Request to add calendar item (incl Auth Token)
Server A
Server B
Success
Calendar Event Added
Success