

AUTH: DATA FLOW

Not web security!

AAA

- **Authentication**

- “this person is who they say they are”

- **Authorization**

- “this person is allowed to do X, Y, Z”

- **Accounting**

- “who the heck is using all our bandwidth?”

AAA

- **Authentication**

- “this person is who they say they are”

- **Authorization**

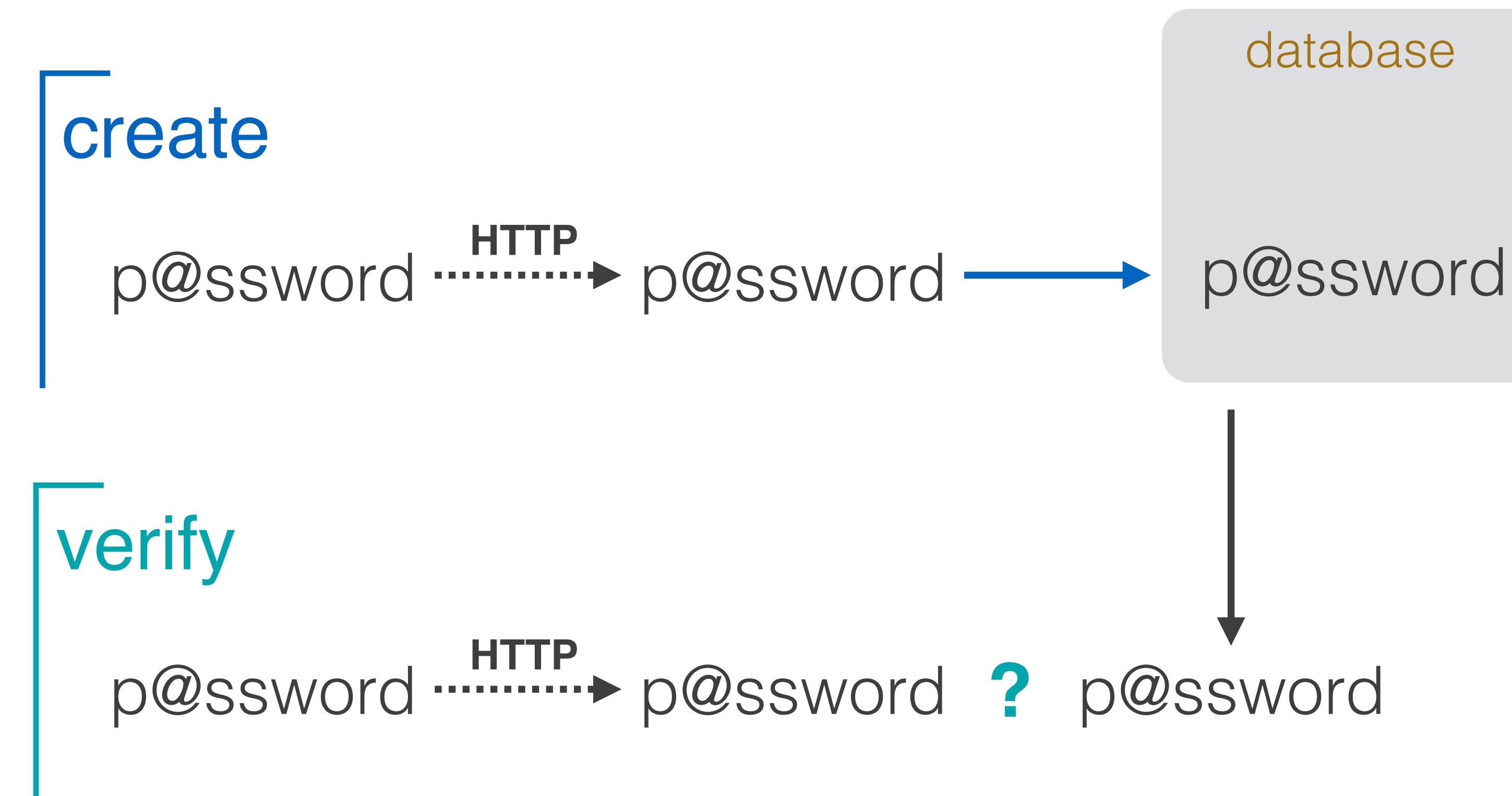
- “this person is allowed to do X, Y, Z”

- **Accounting**

- “who the heck is using all our bandwidth?”



LOGIN/SIGNUP



STAYING LOGGED IN



THE PROBLEM

What if we want to store some information about each client/server relationship (session)?



IDEAS?



IDEAS?

- script variables



IDEAS?

- script variables
- database documents



SCRIPT VARIABLES?



SCRIPT VARIABLES?

Data will not persist across browser pages



DATABASE DOCUMENTS?



DATABASE DOCUMENTS?

*Yes, but login credentials would have to be sent
with EVERY request*



cookie!



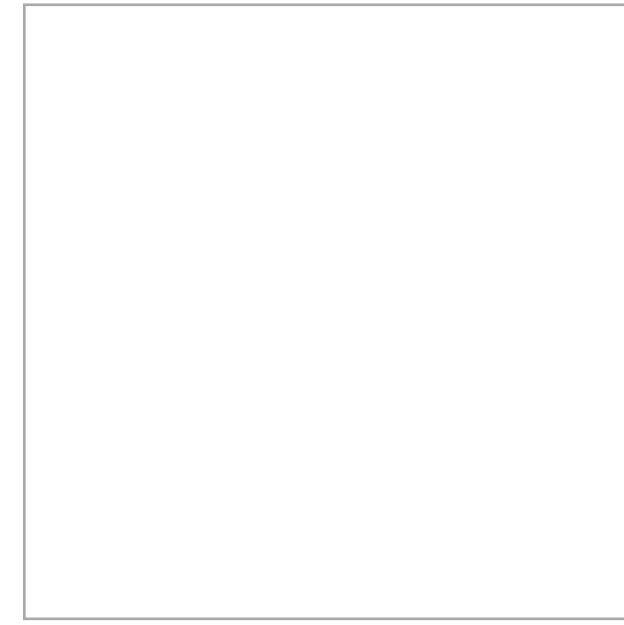


= small text file

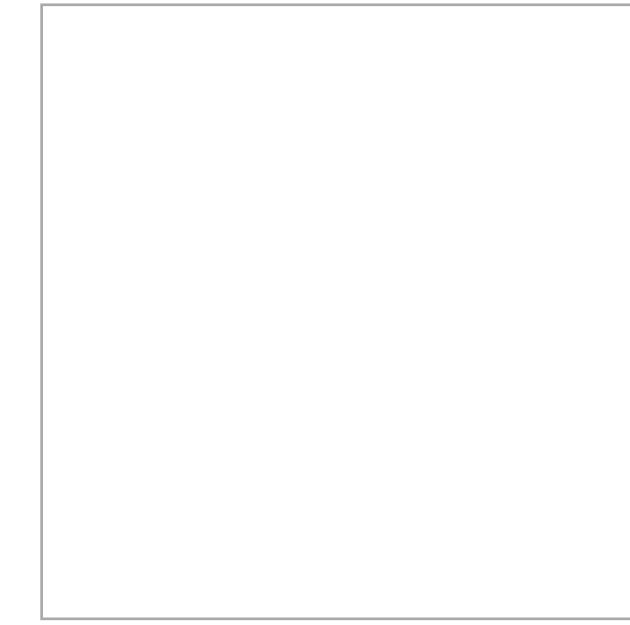


HTTP

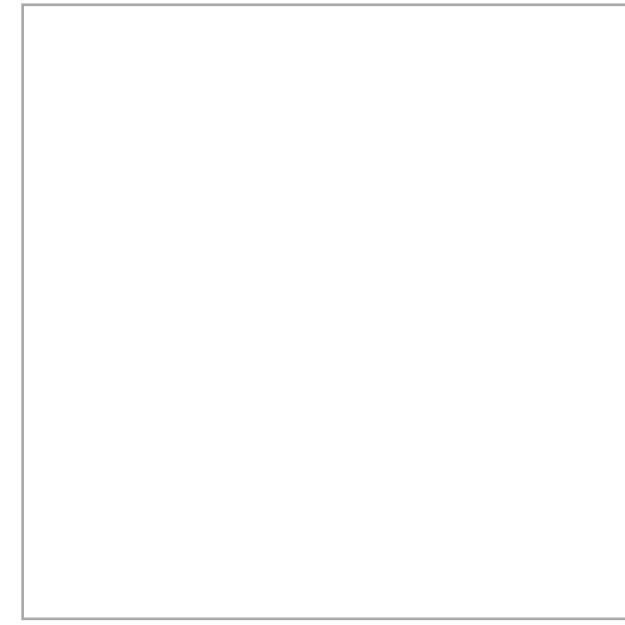
client



“the internet”



server





HTTP

client

request

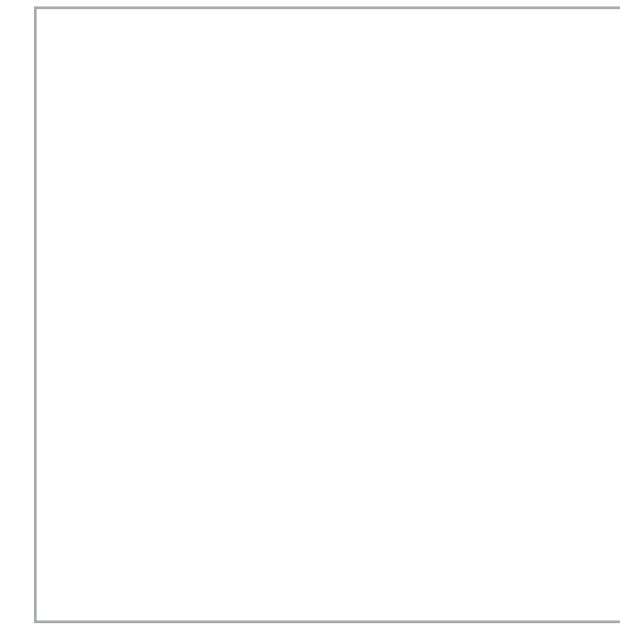
“the internet”

server

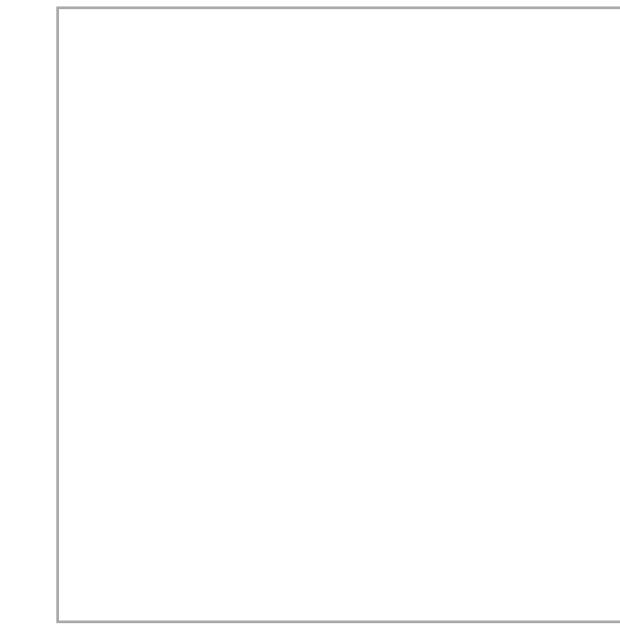


HTTP

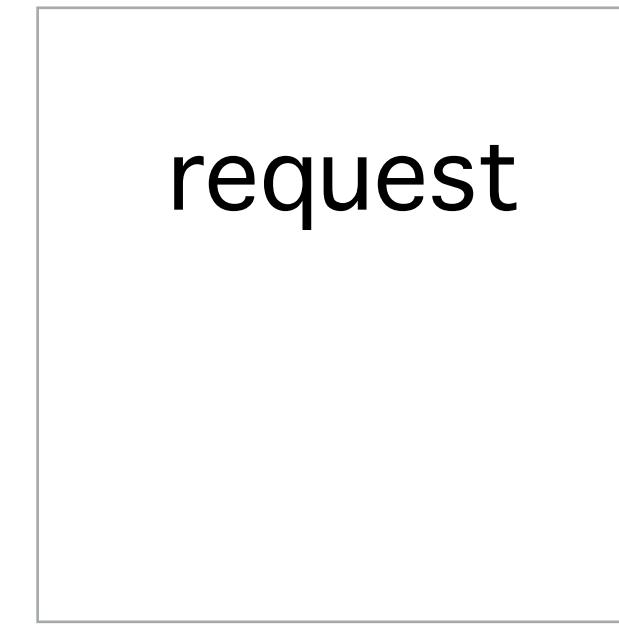
client



“the internet”



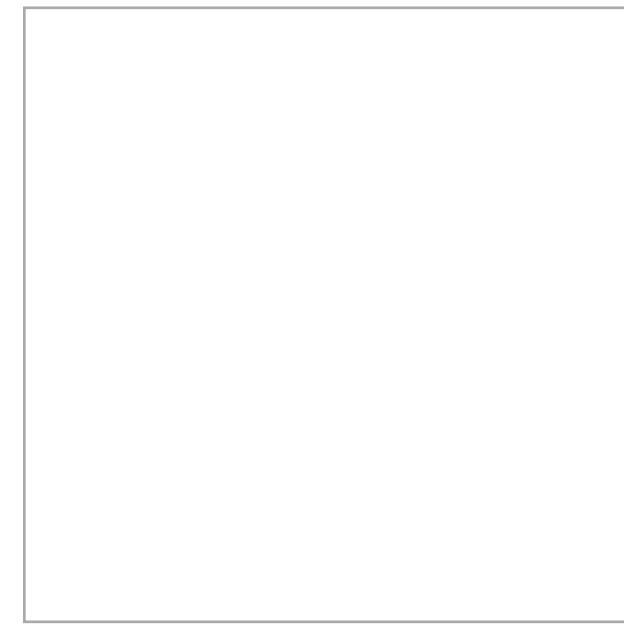
server



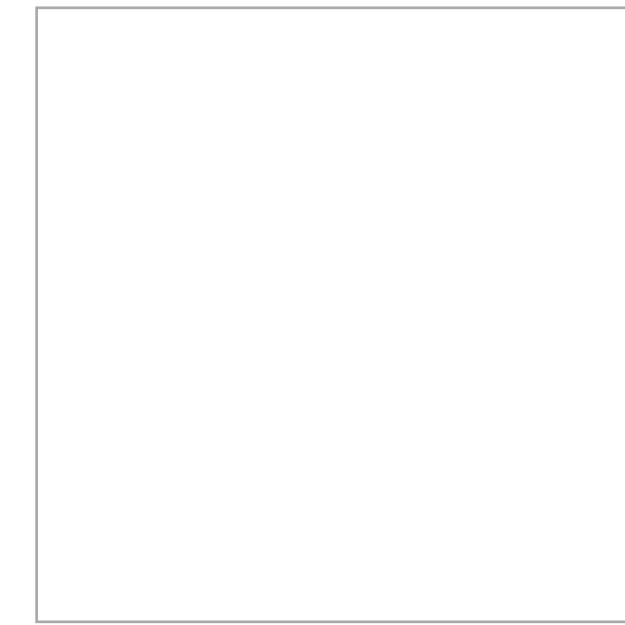


HTTP

client



“the internet”



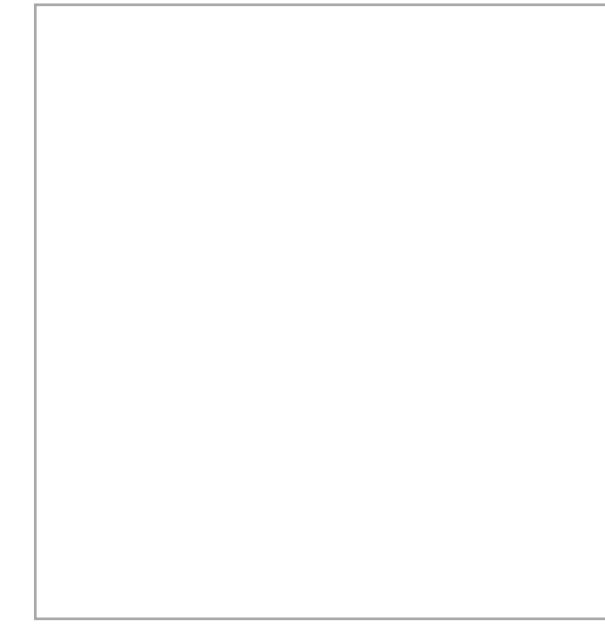
server



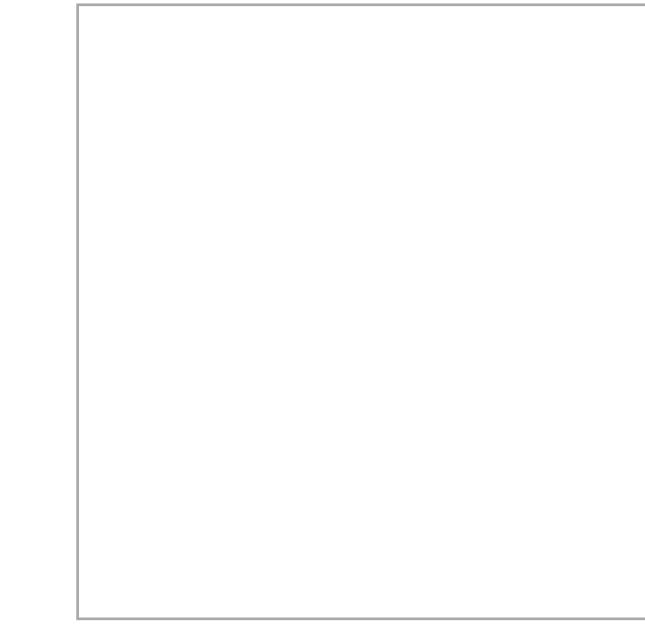


HTTP

client



“the internet”



server





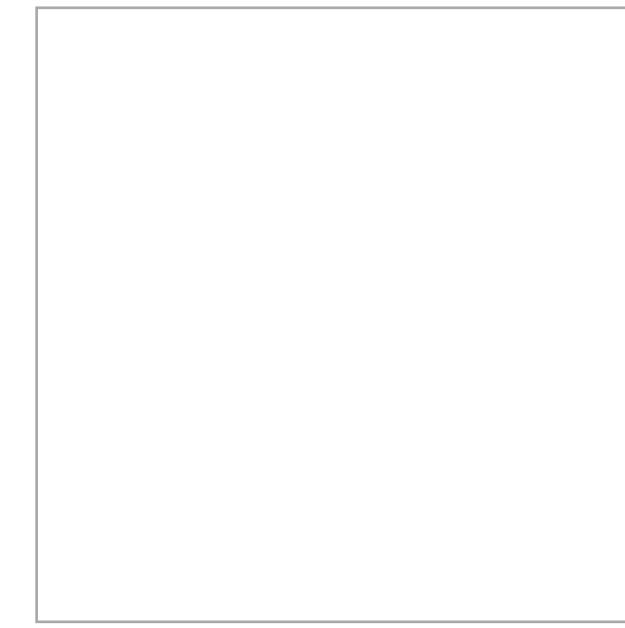
HTTP

client

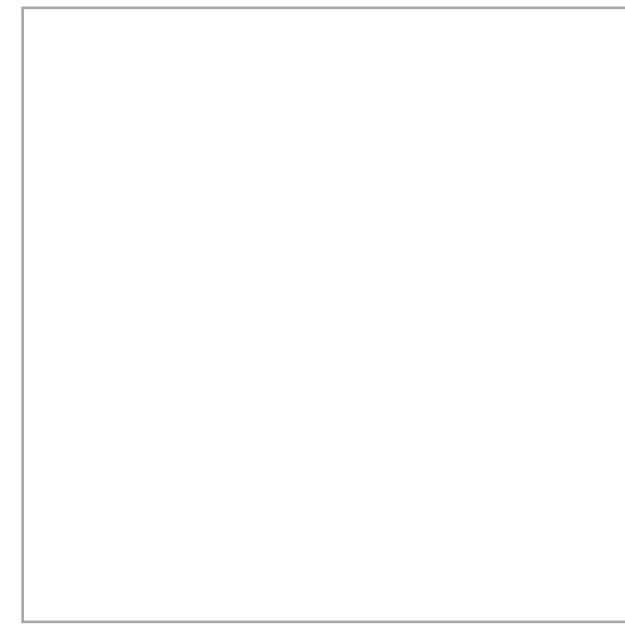


response

“the internet”



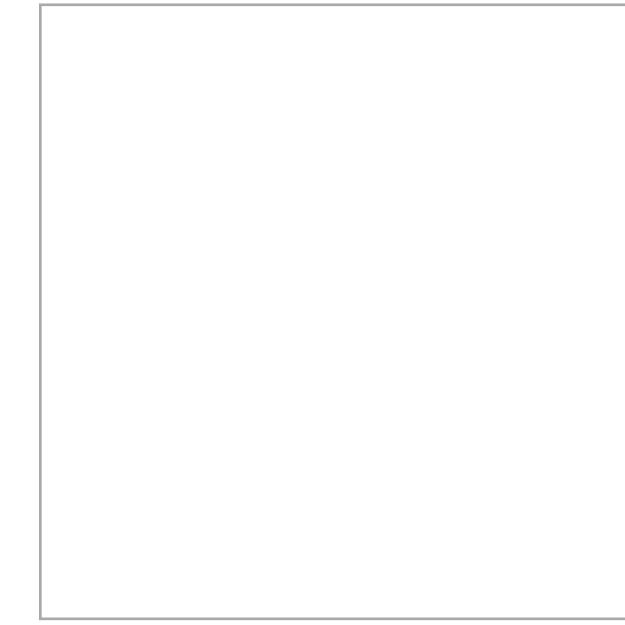
server



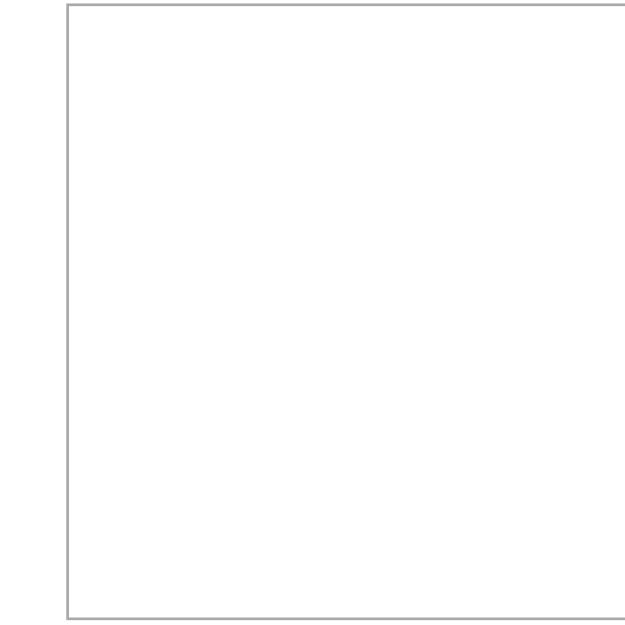


WITH COOKIES

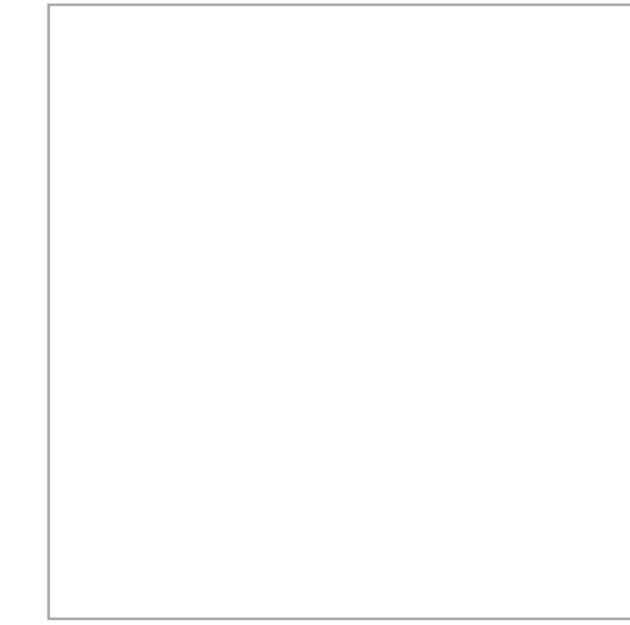
client



“the internet”

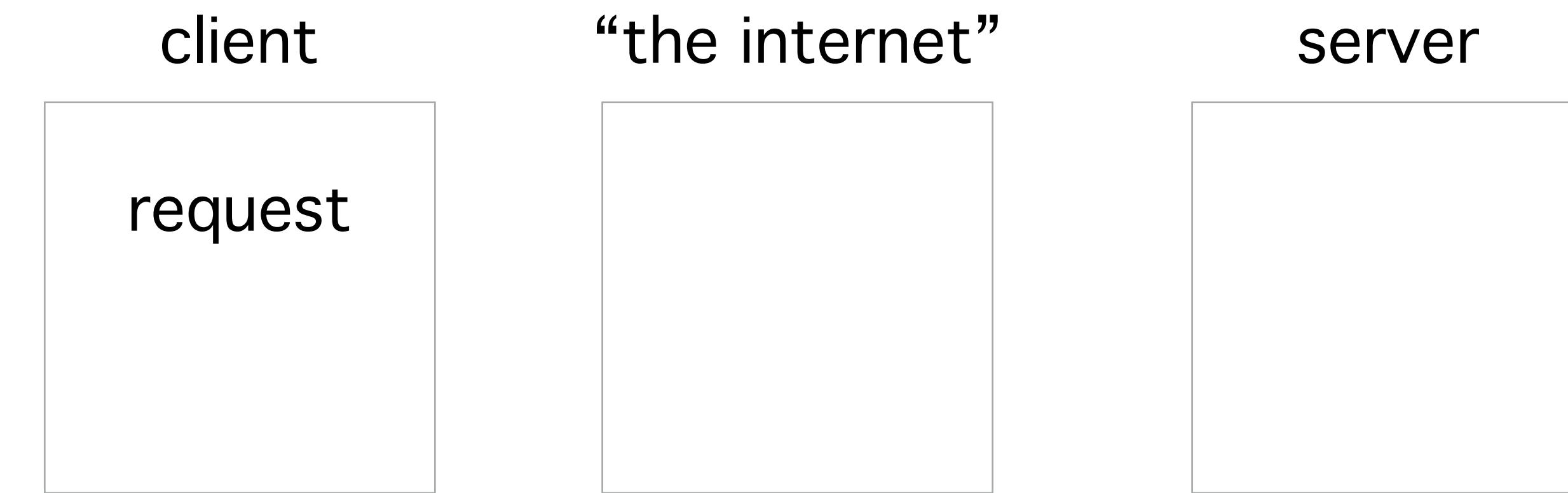


server





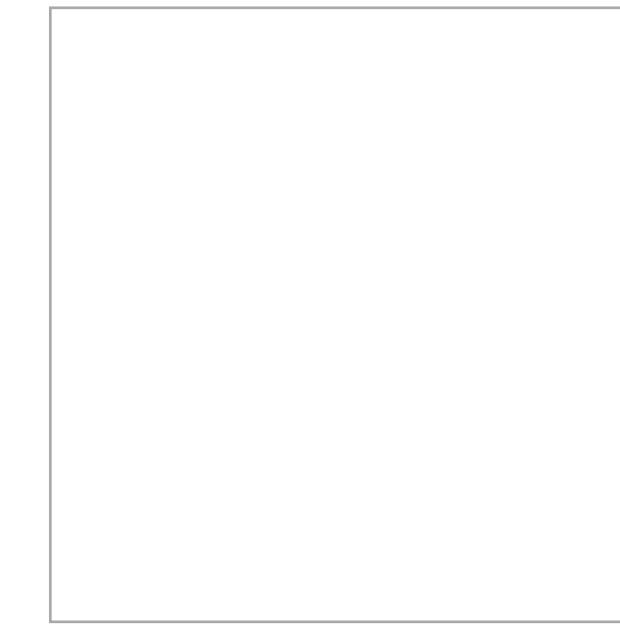
WITH COOKIES



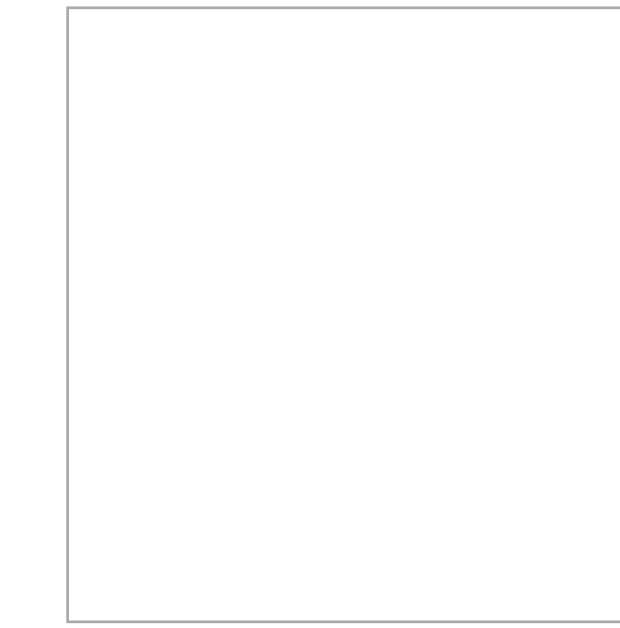


WITH COOKIES

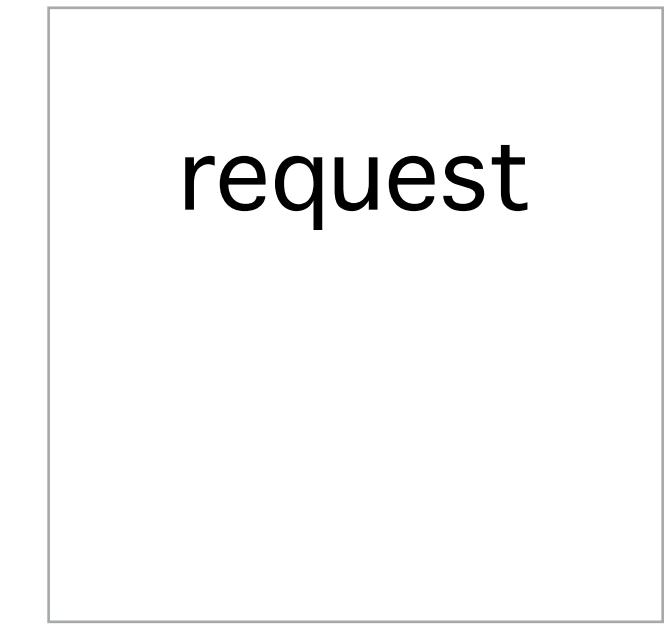
client



“the internet”



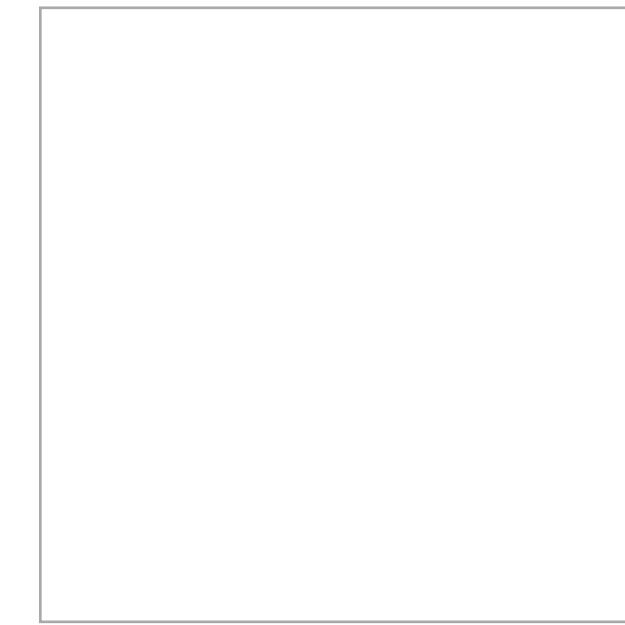
server



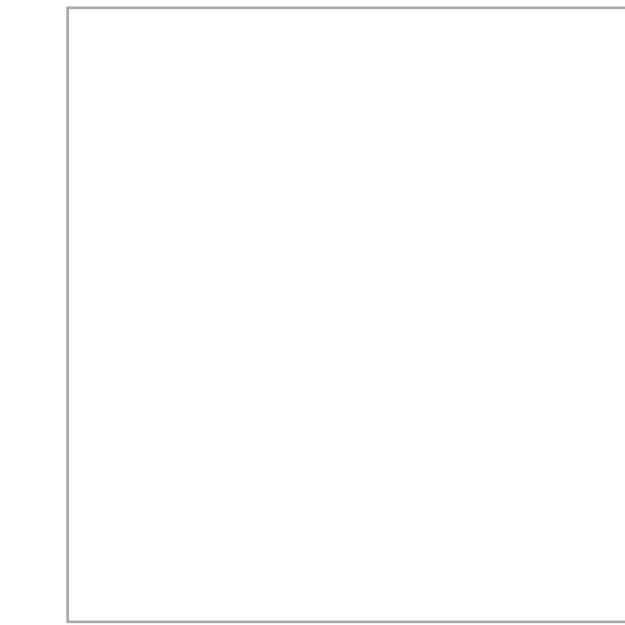


WITH COOKIES

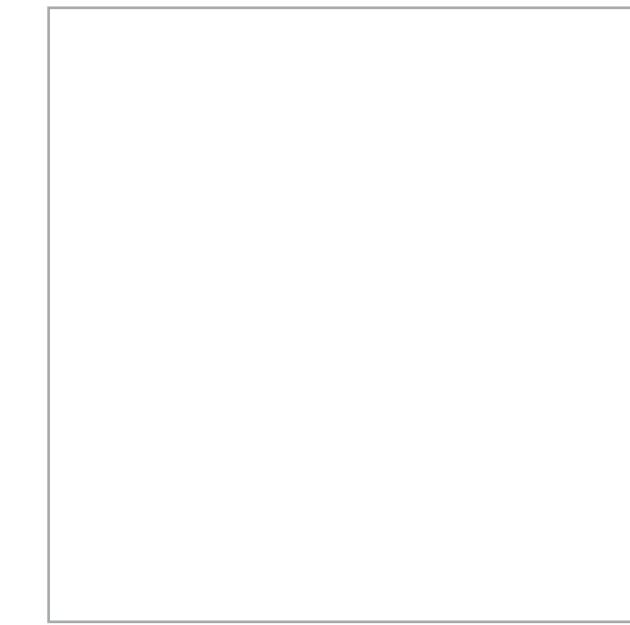
client



“the internet”

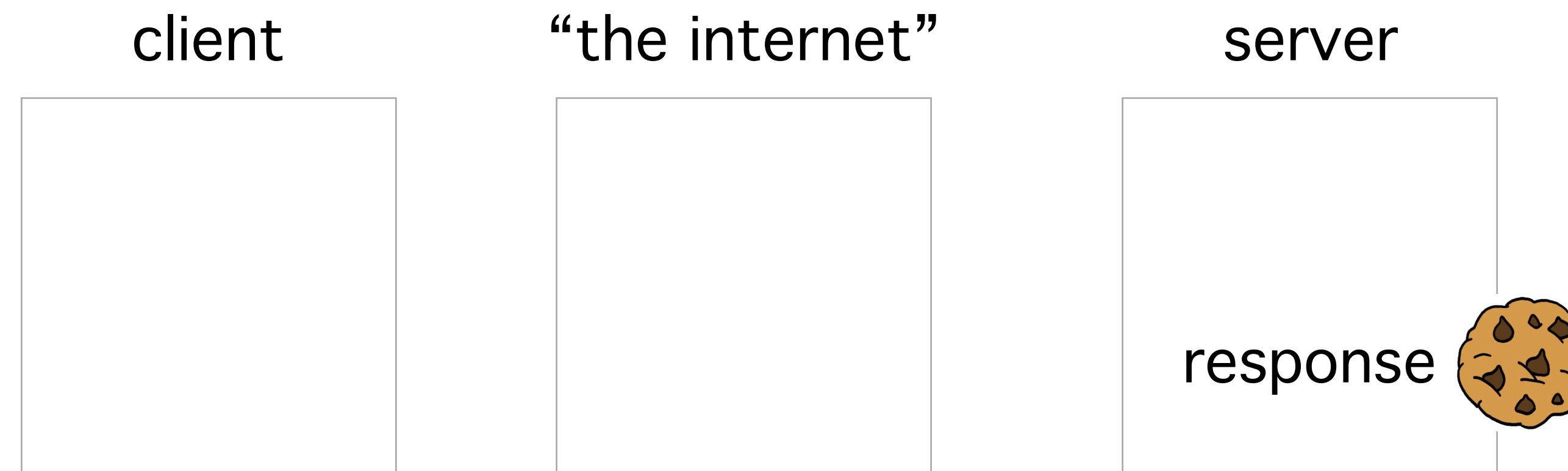


server



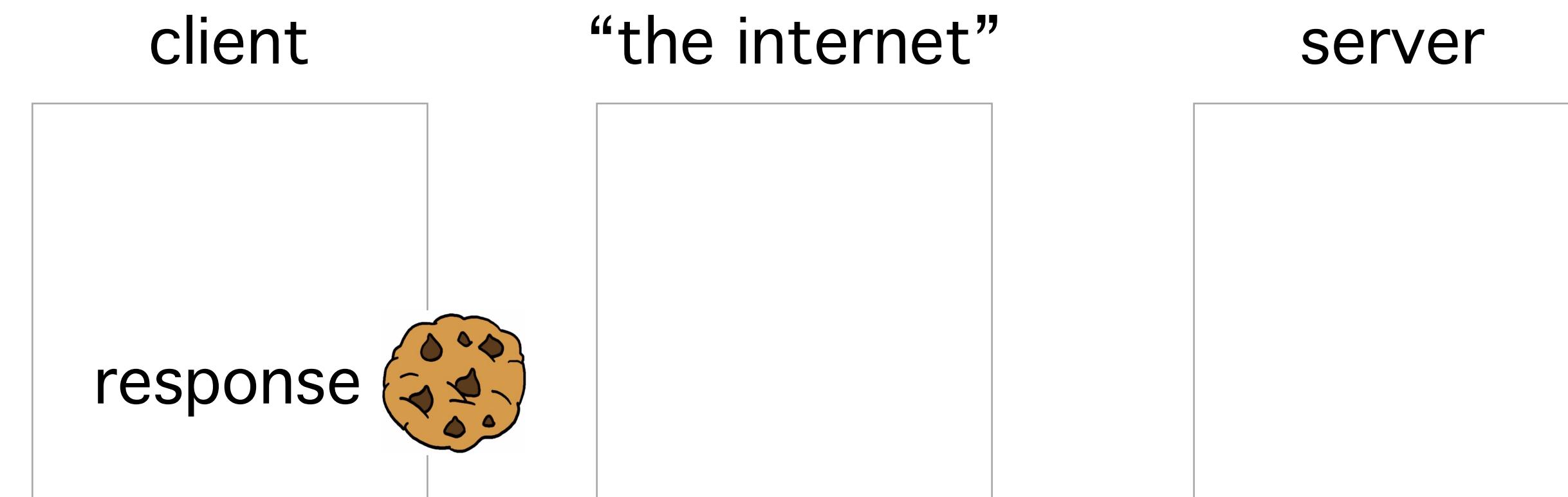


WITH COOKIES





WITH COOKIES



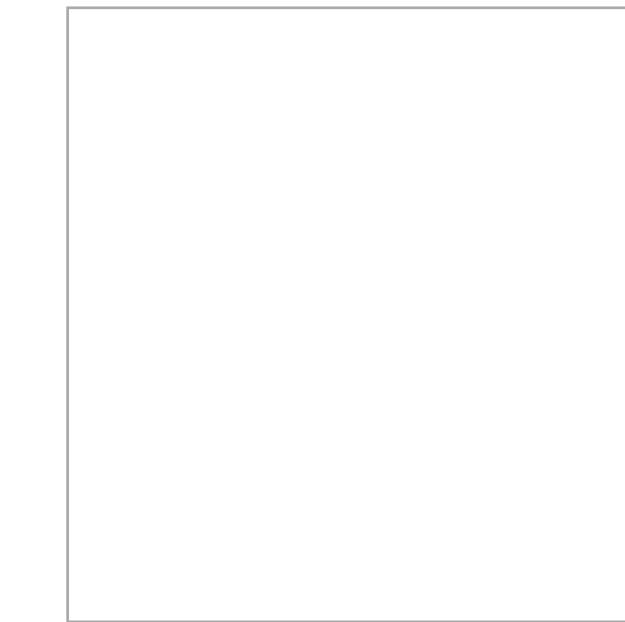


WITH COOKIES

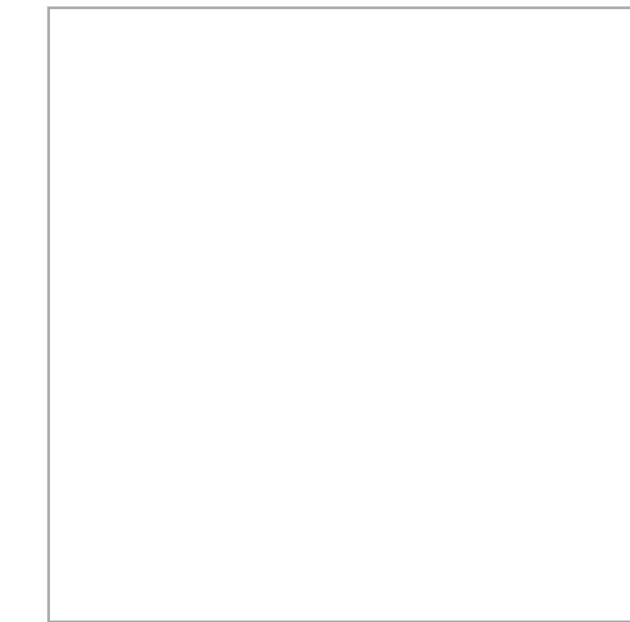
client



“the internet”

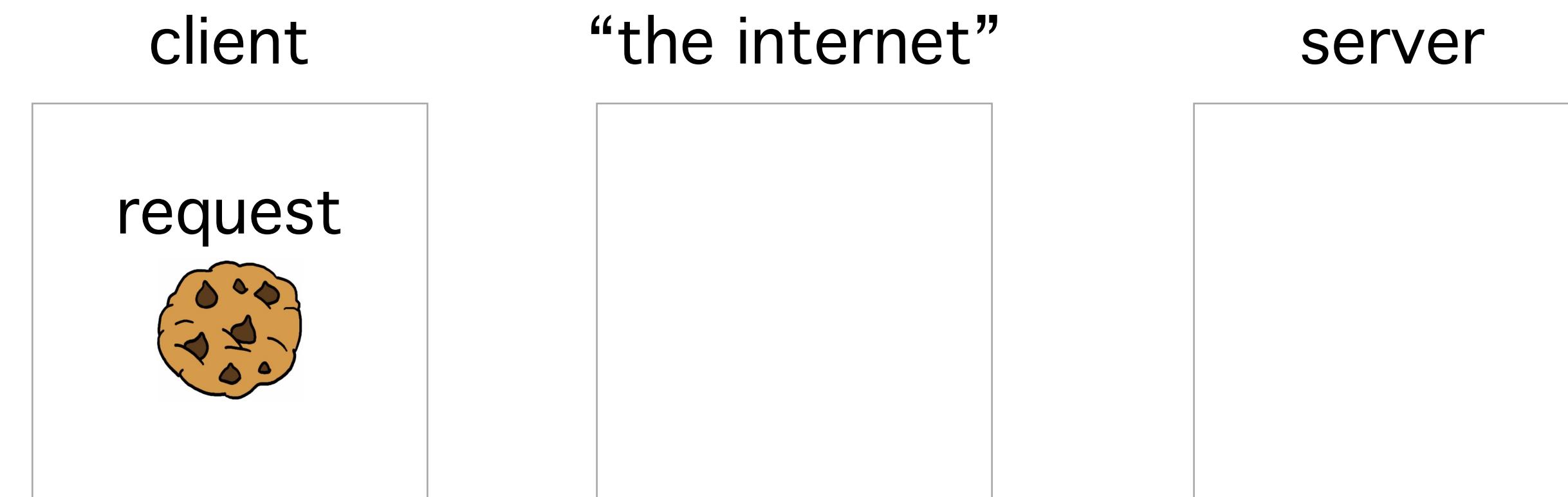


server



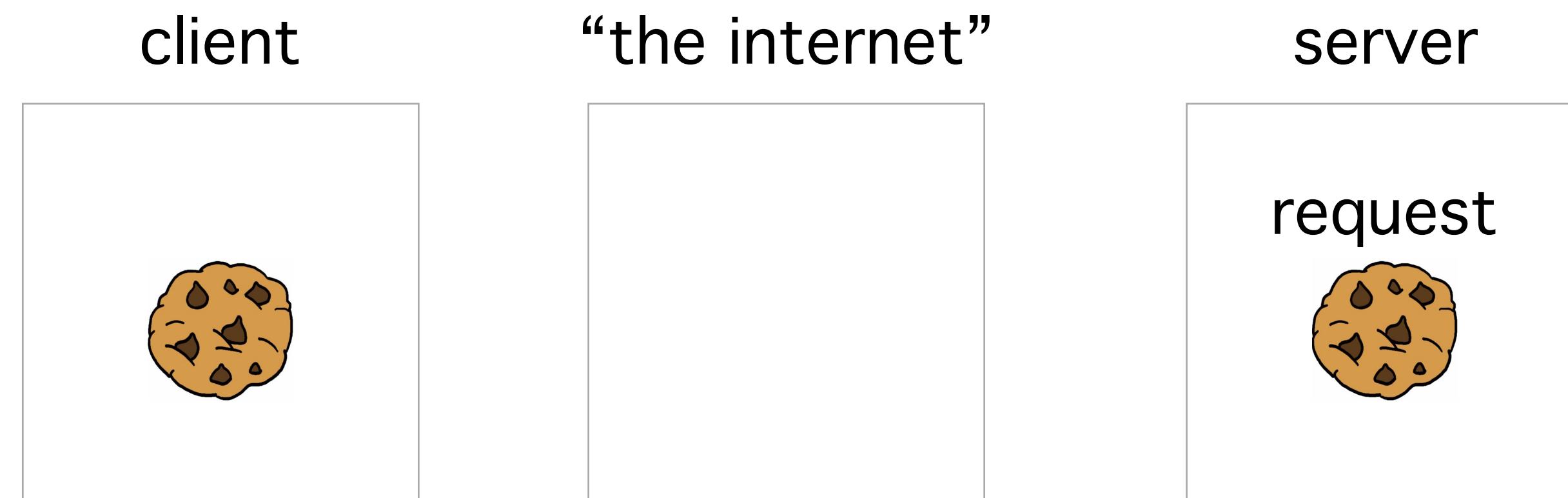


WITH COOKIES



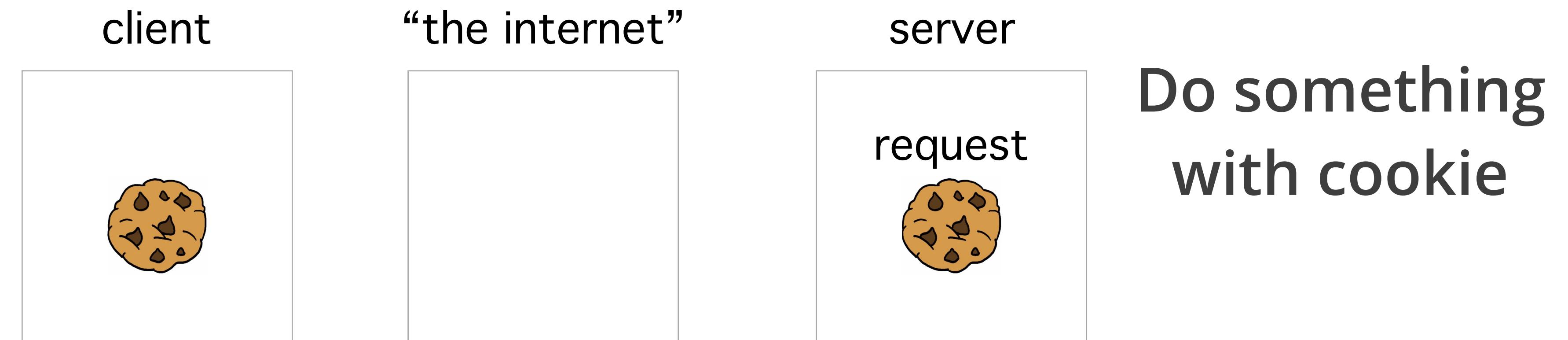


WITH COOKIES





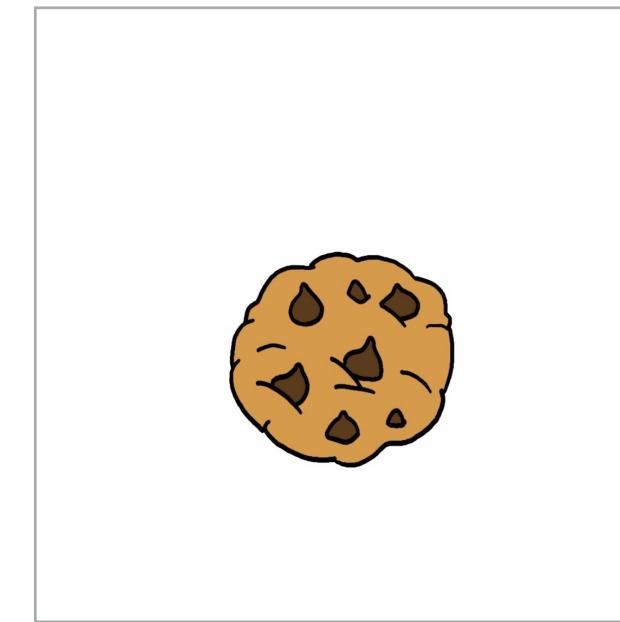
WITH COOKIES



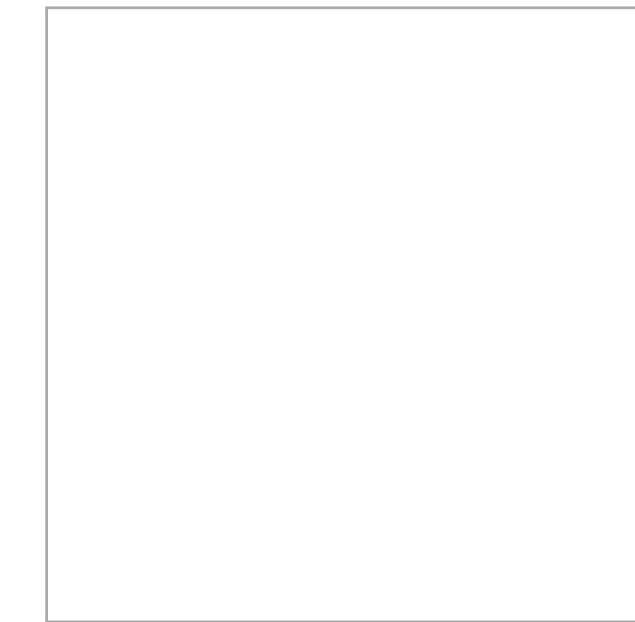


WITH COOKIES

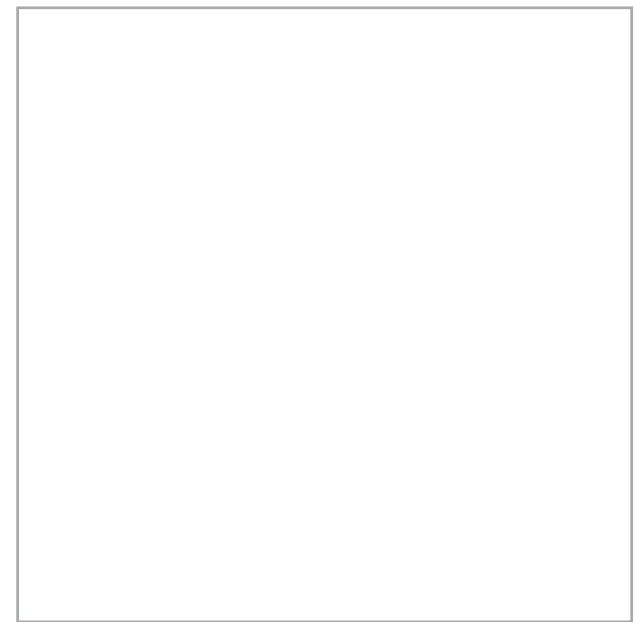
client



“the internet”



server



**Do something
with cookie**

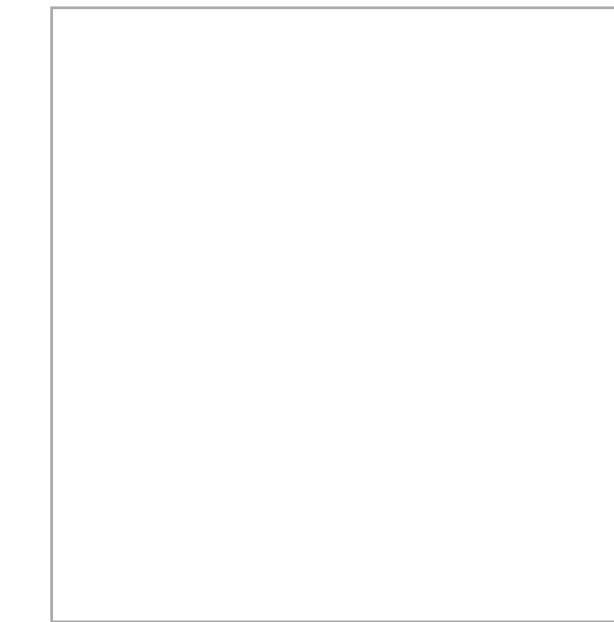


WITH COOKIES

client



“the internet”



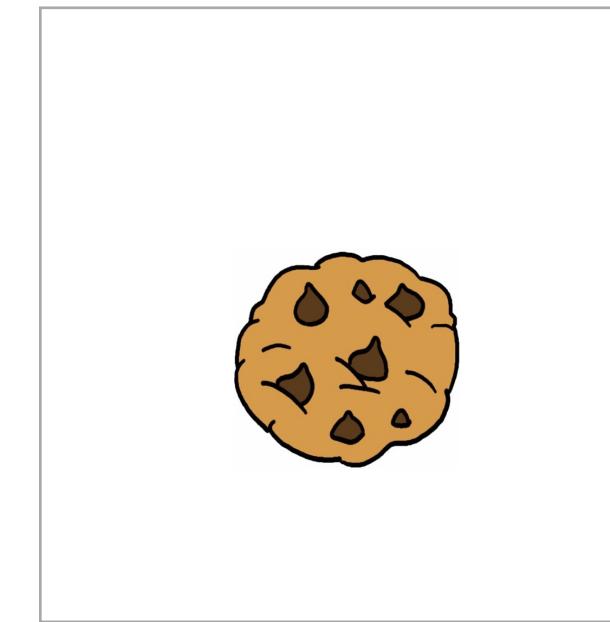
server



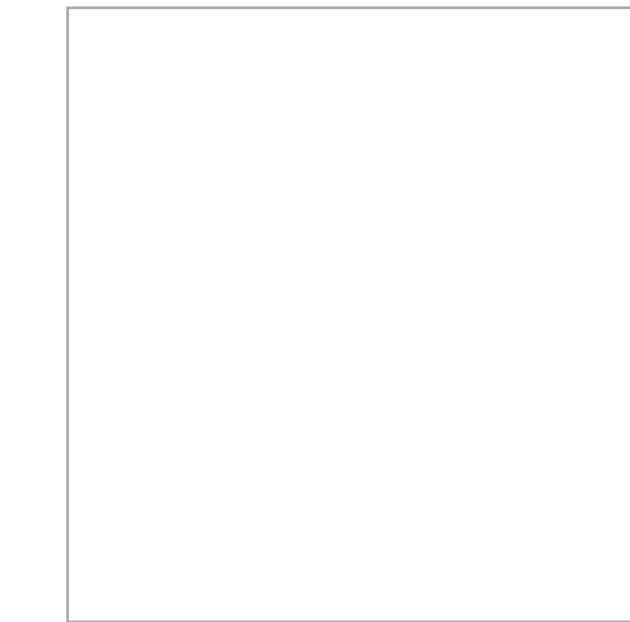


WITH COOKIES

client



“the internet”



server



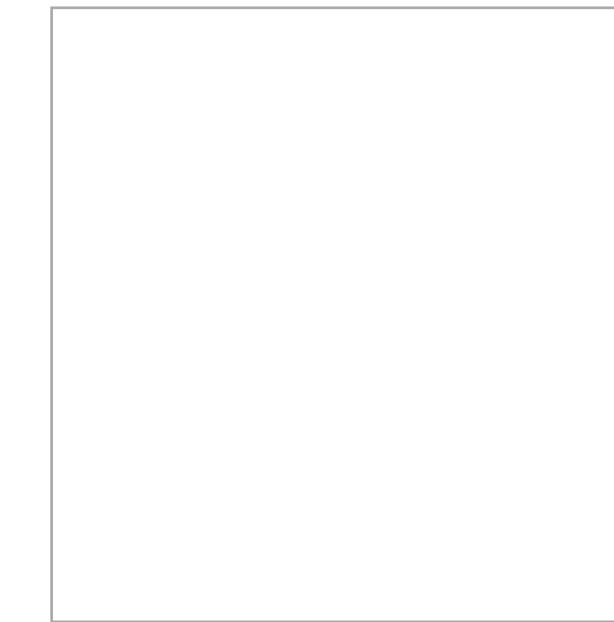


WITH COOKIES

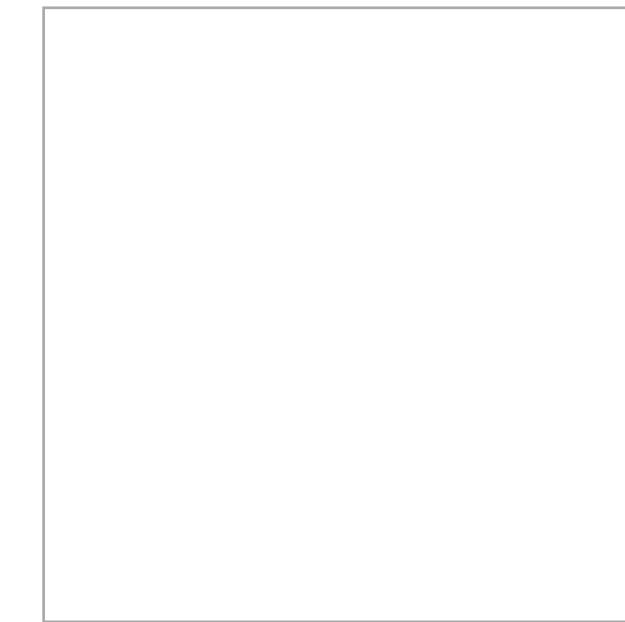
client



“the internet”



server





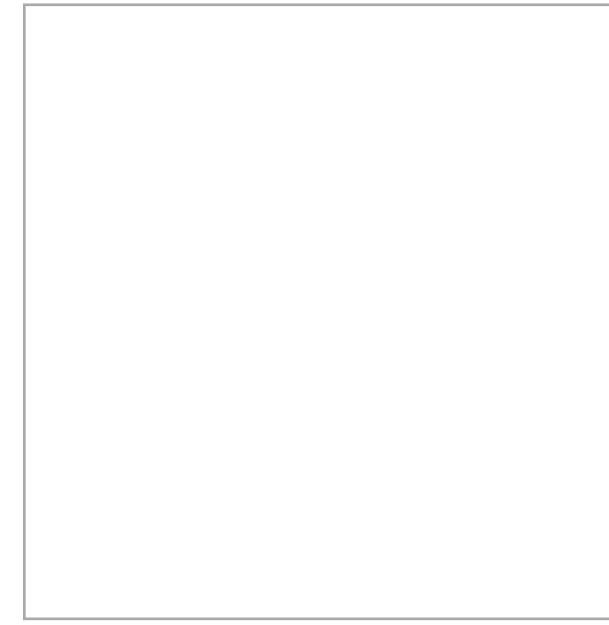
COOKIES & SESSIONS

No need to authenticate for every request

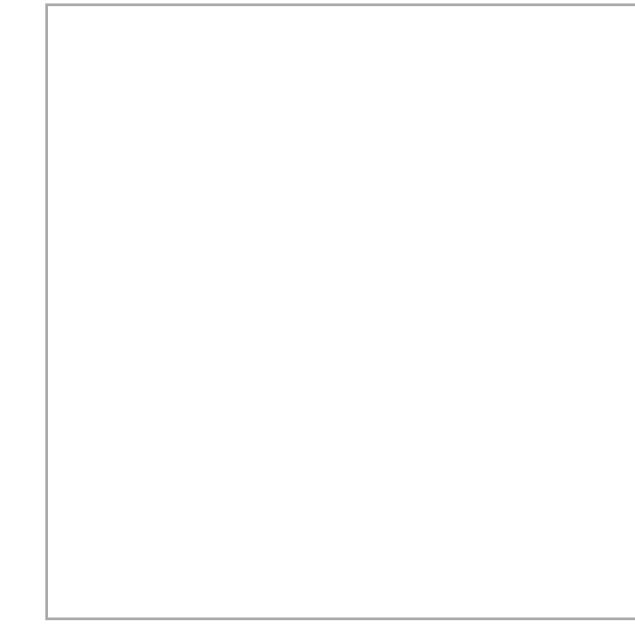


SESSIONS

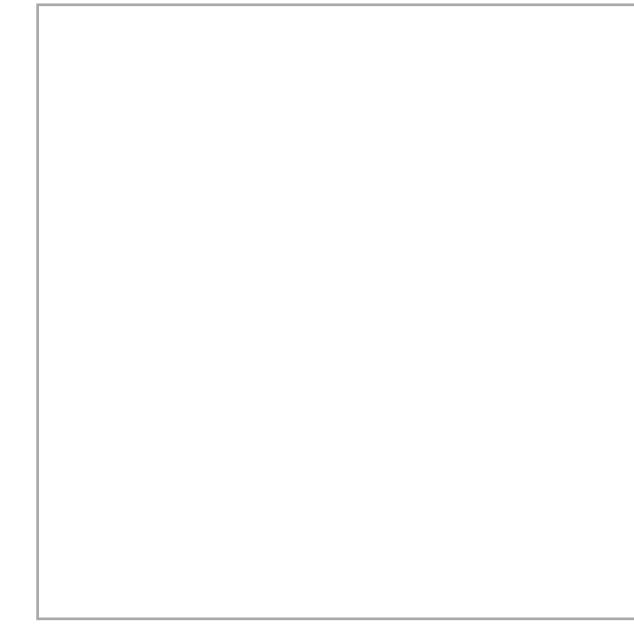
client



“the internet”



server





SESSIONS

client

request

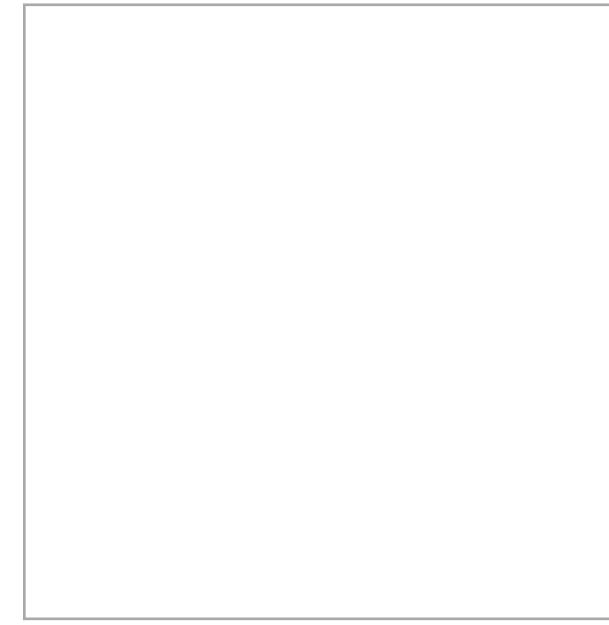
“the internet”

server

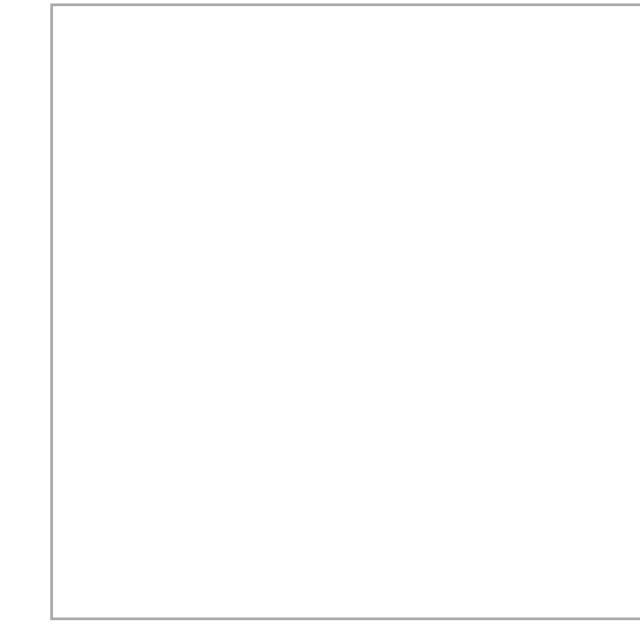


SESSIONS

client



“the internet”



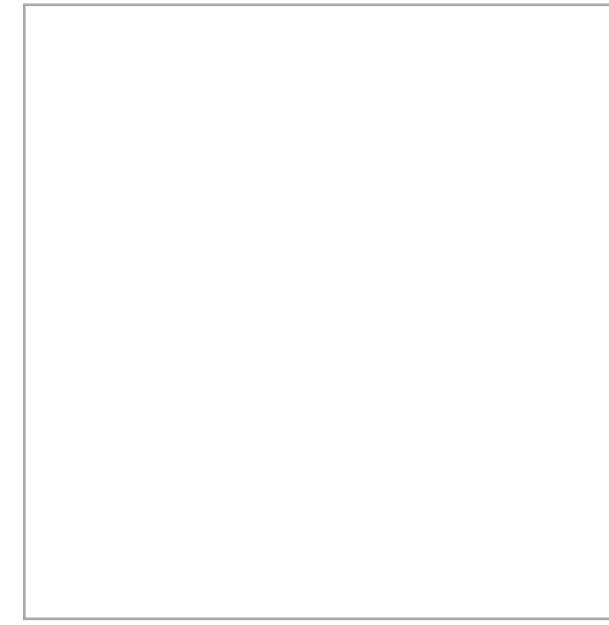
server



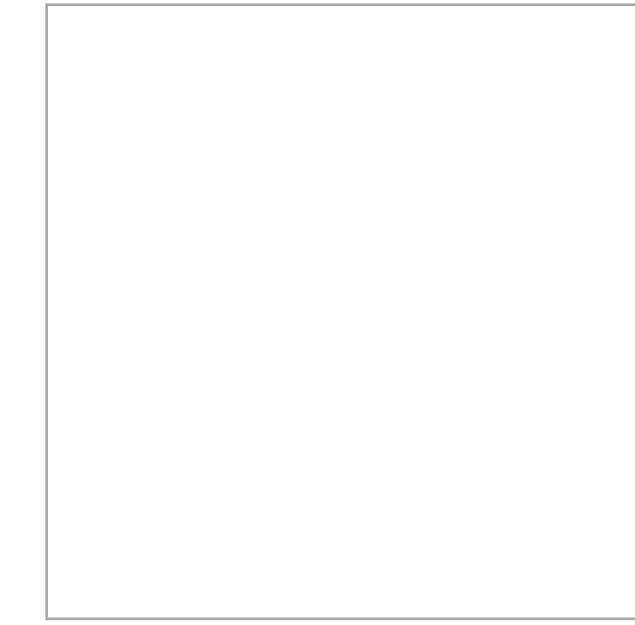


SESSIONS

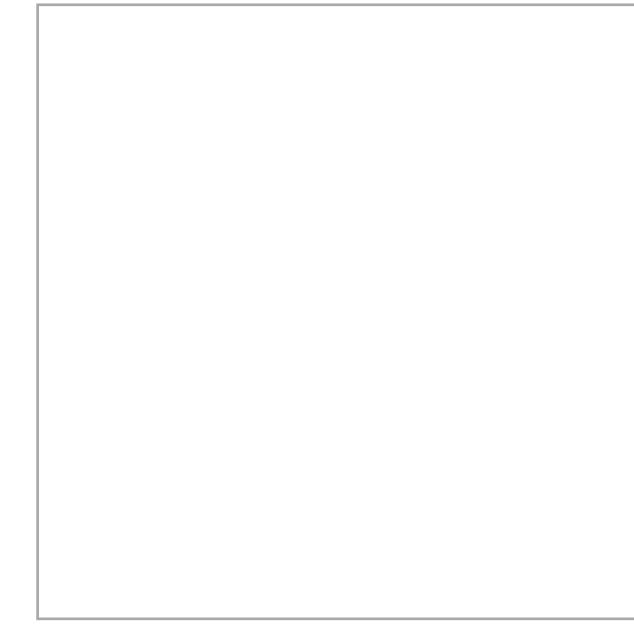
client



“the internet”



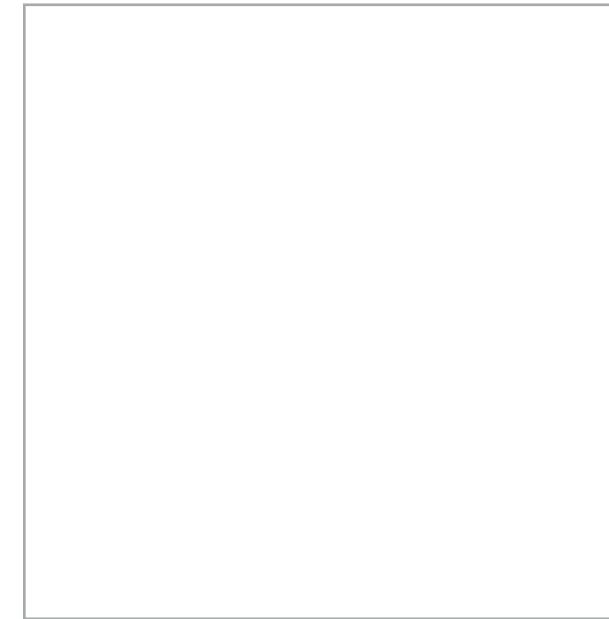
server





SESSIONS

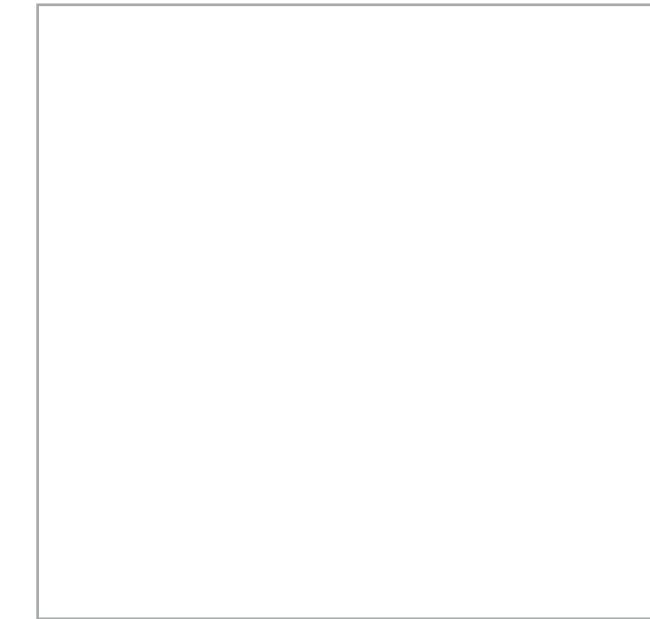
client



“the internet”



server

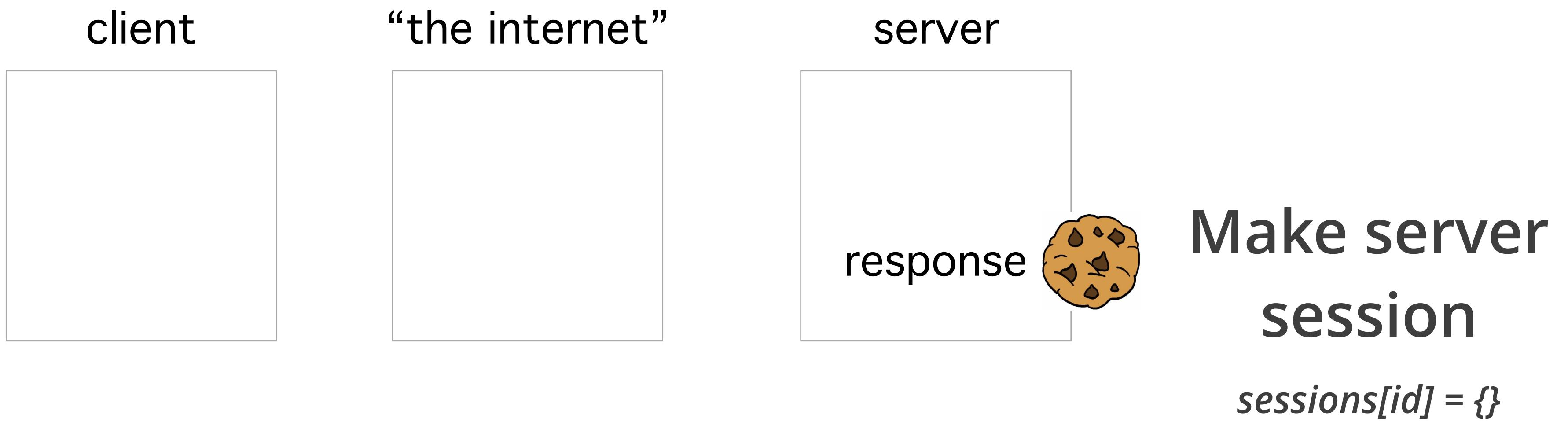


**Make server
session**

sessions[id] = {}



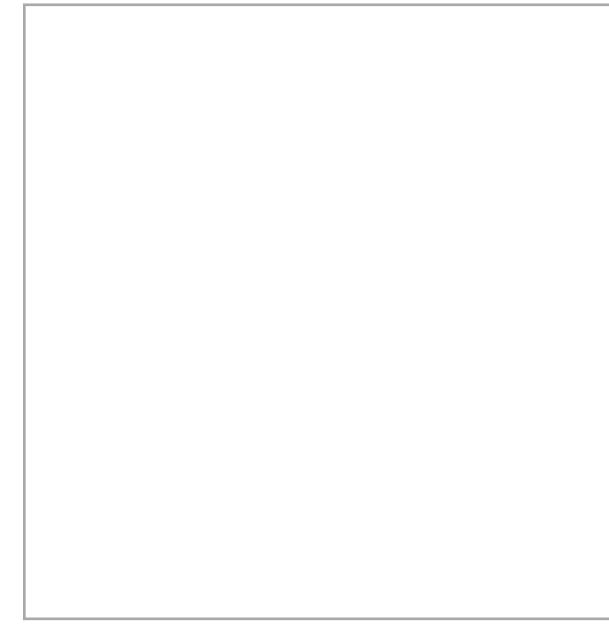
SESSIONS



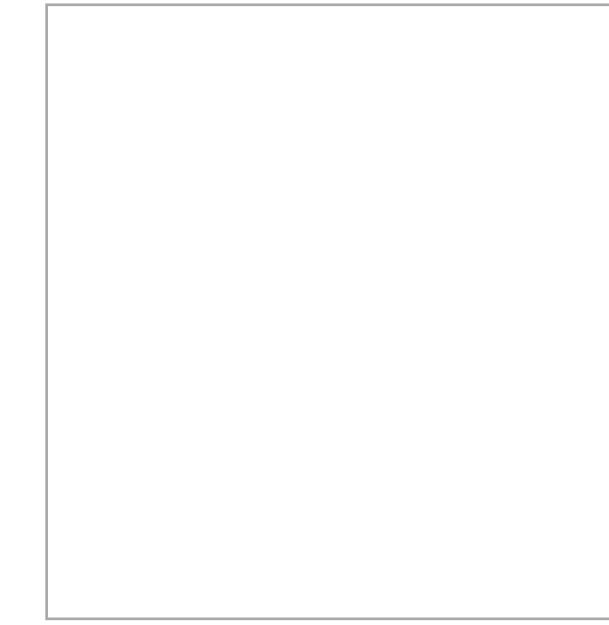


SESSIONS

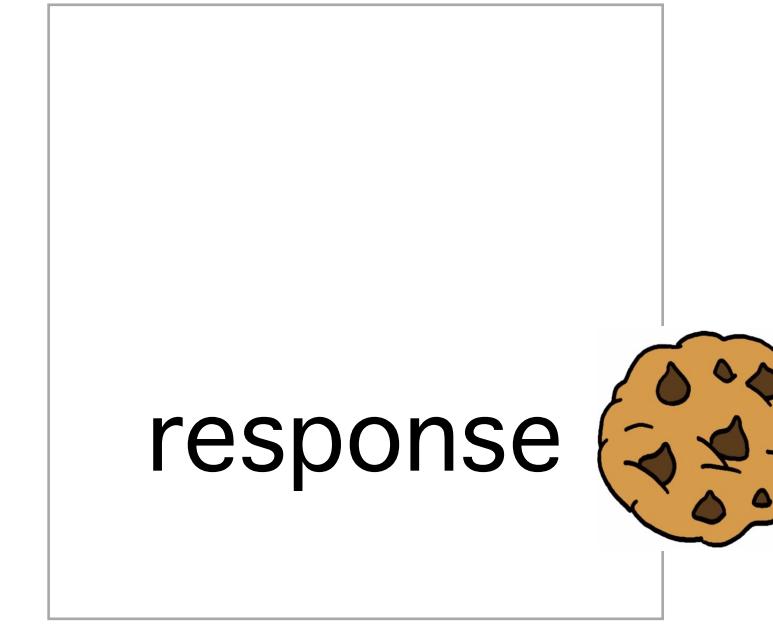
client



“the internet”

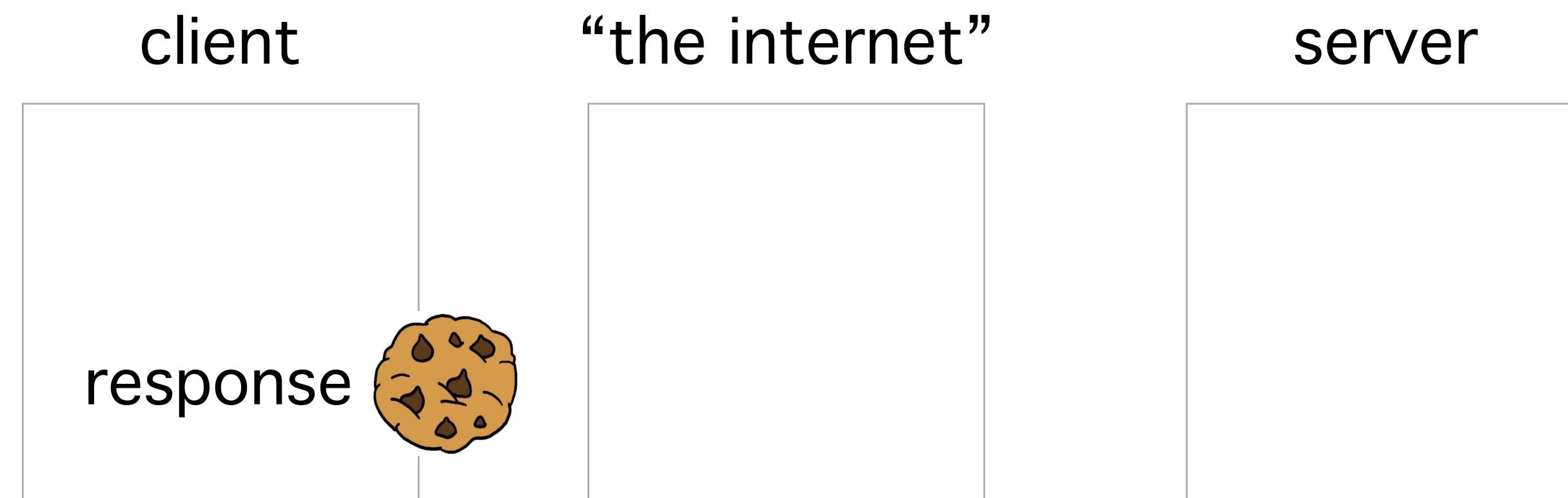


server





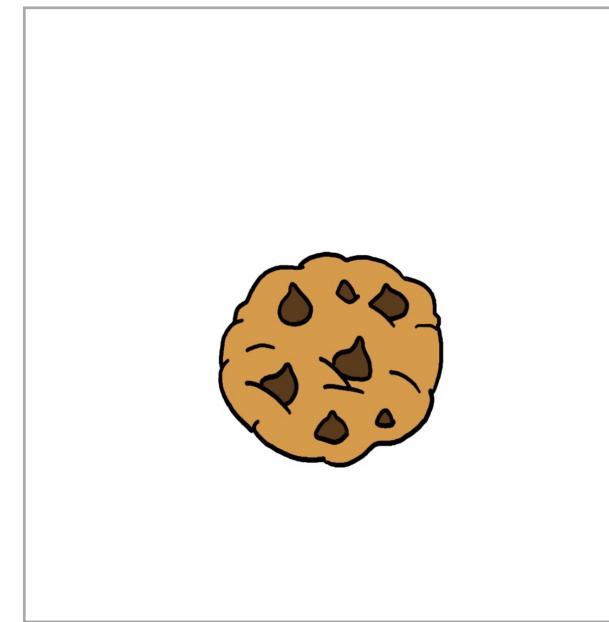
SESSIONS



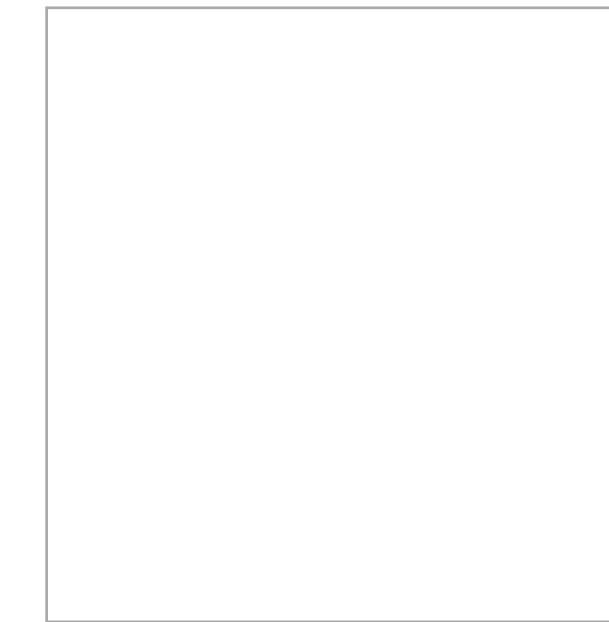


SESSIONS

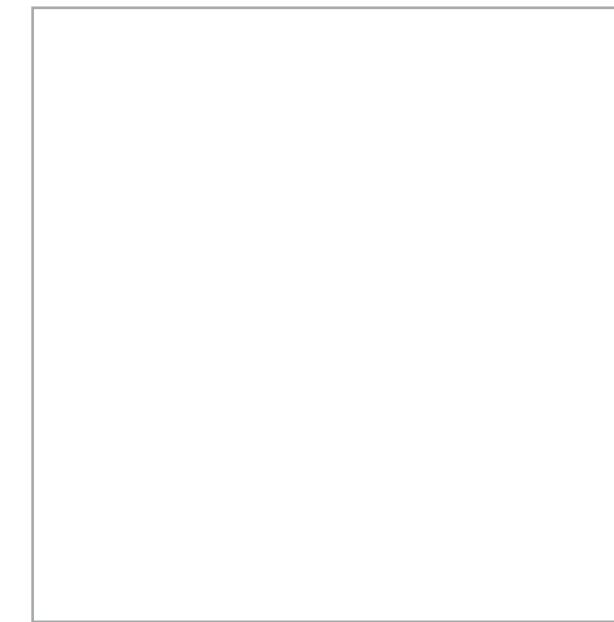
client



“the internet”



server

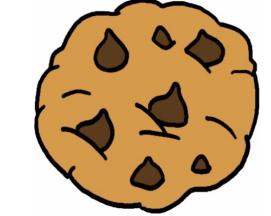




SESSIONS

client

request



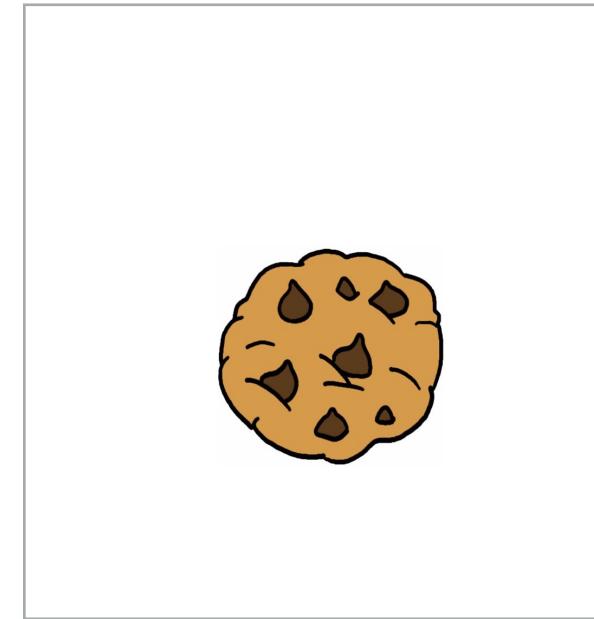
“the internet”

server



SESSIONS

client



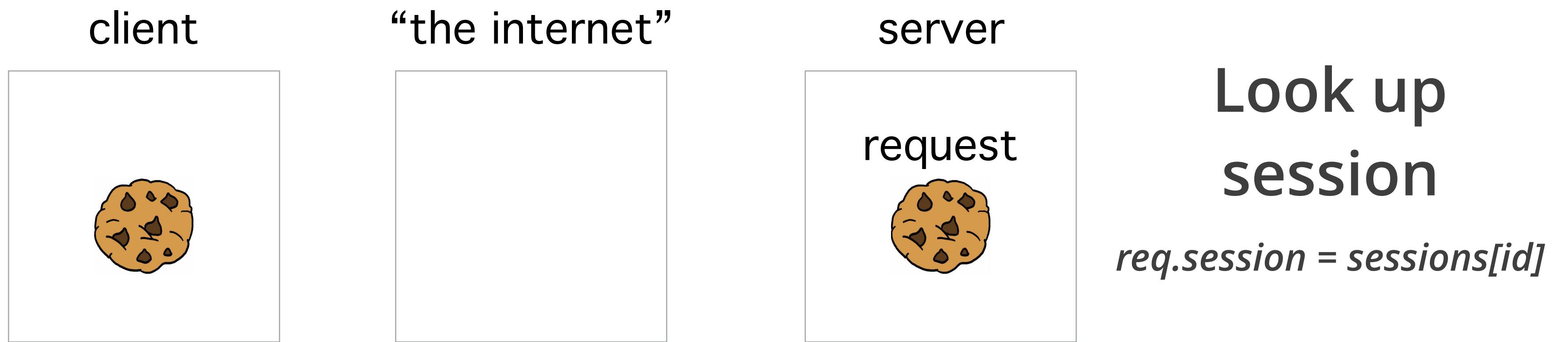
“the internet”

server





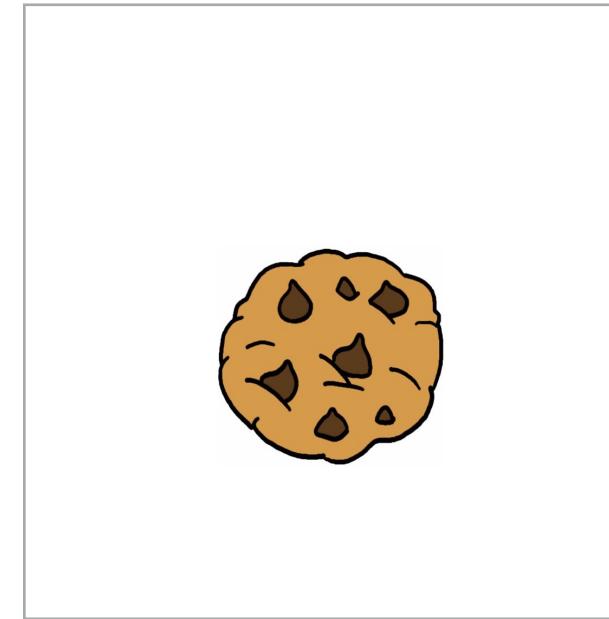
SESSIONS



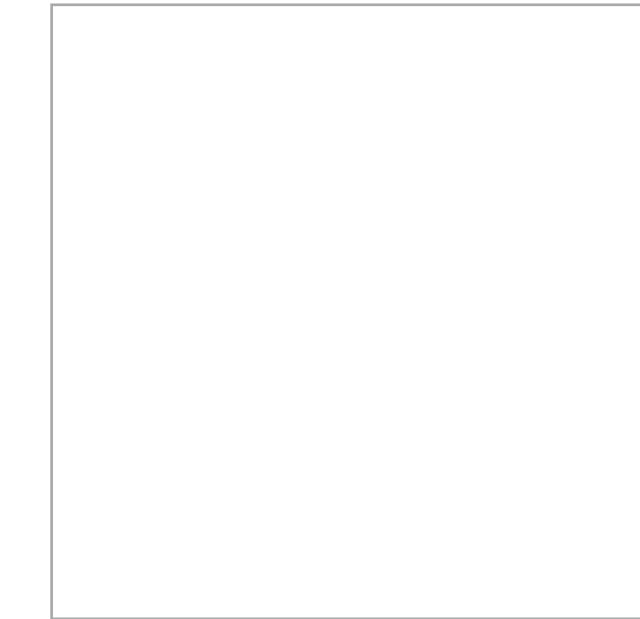


SESSIONS

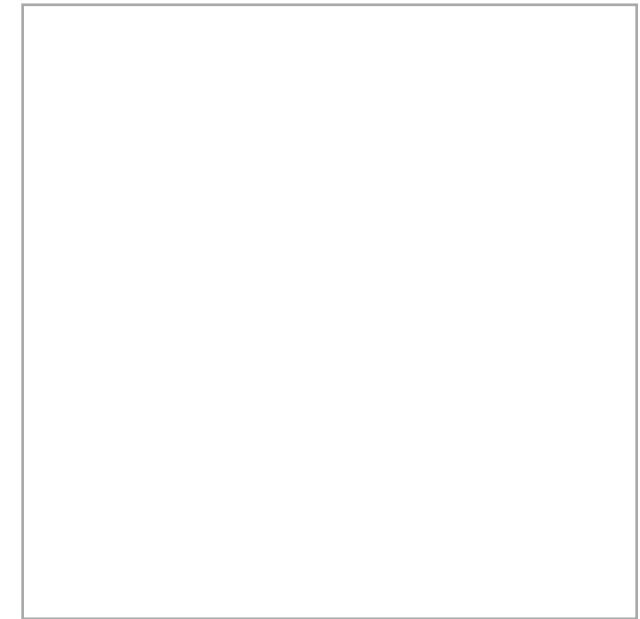
client



“the internet”



server



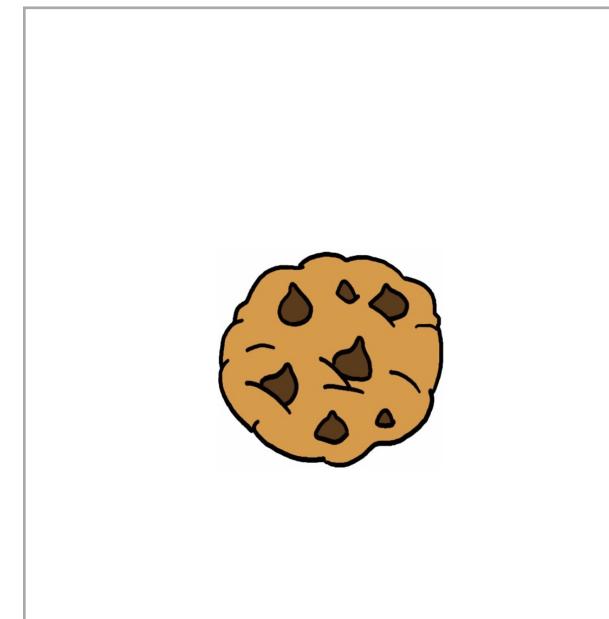
Look up
session

req.session = sessions[id]

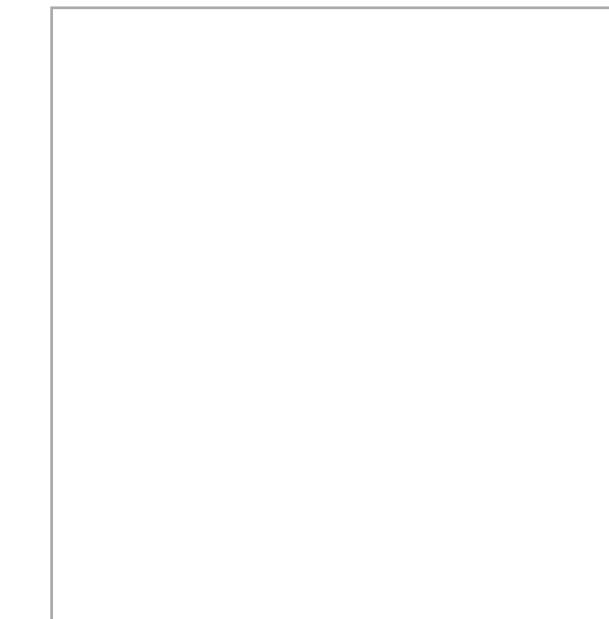


SESSIONS

client



“the internet”



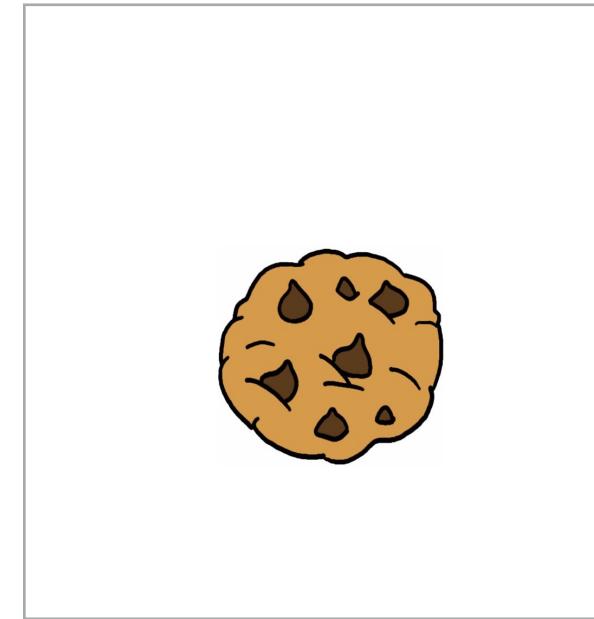
server



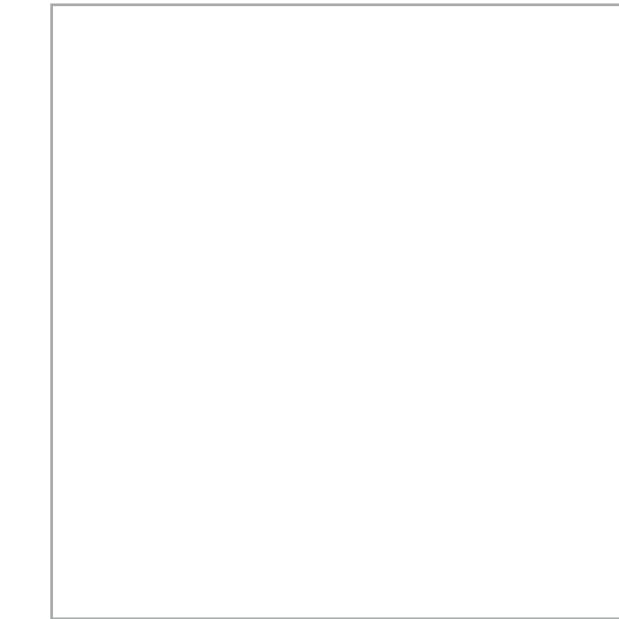


SESSIONS

client



“the internet”



server





SESSIONS

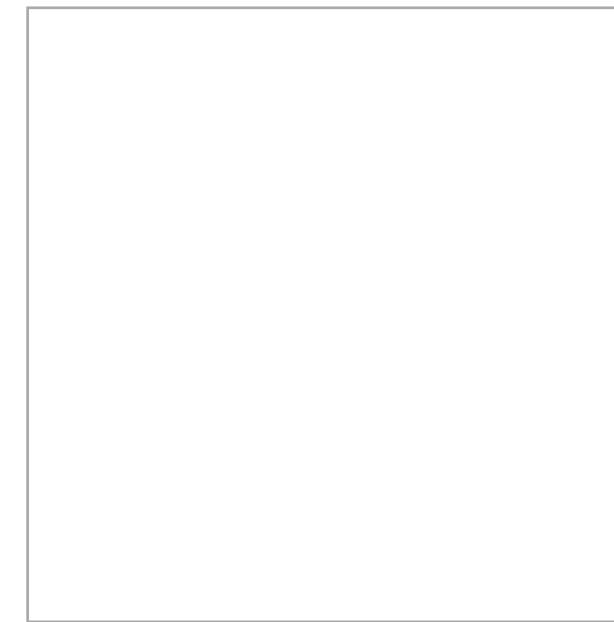
client



“the internet”



server





COOKIES & SESSIONS

- Server gives client a cookie with ID only
- Client keeps cookie and sends with all requests
- Server loads request-specific session (stored in RAM)





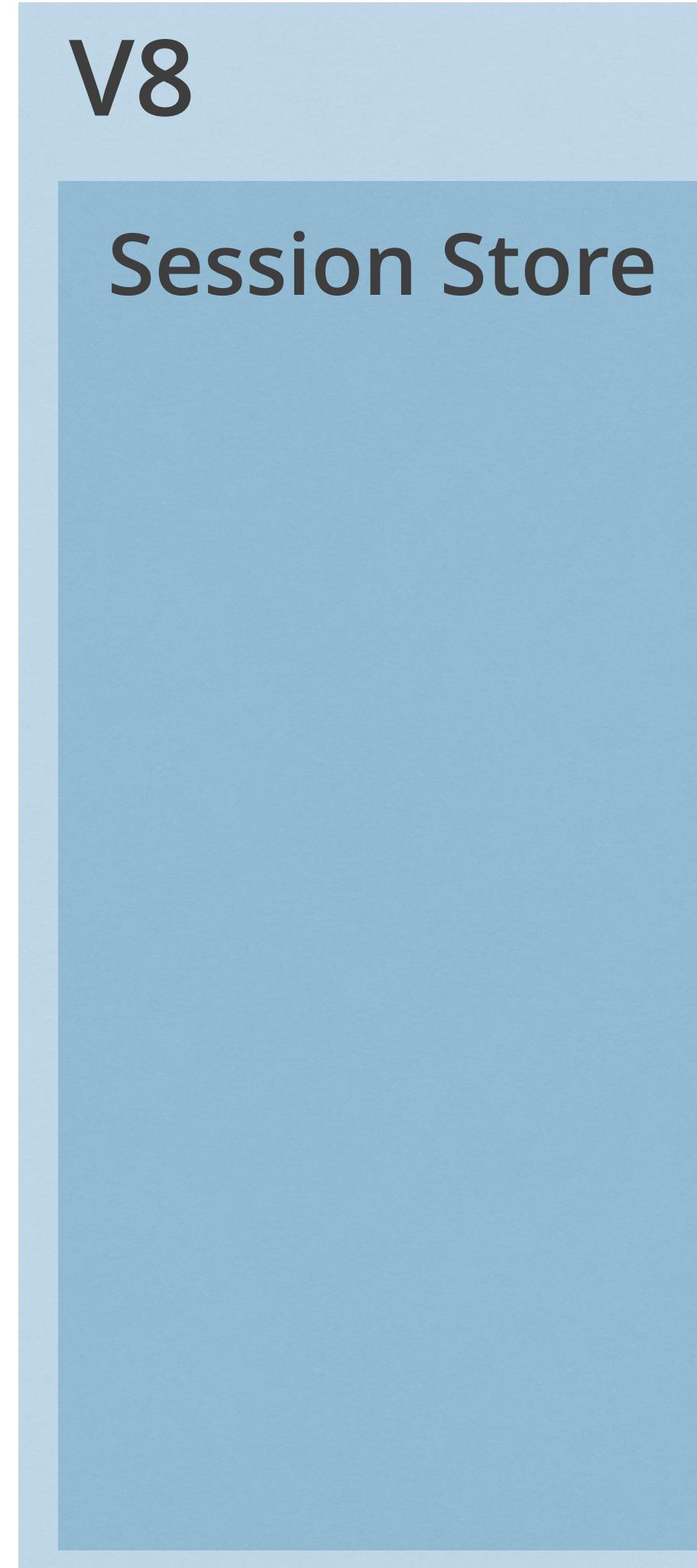
SESSIONS

SESSIONS
IN



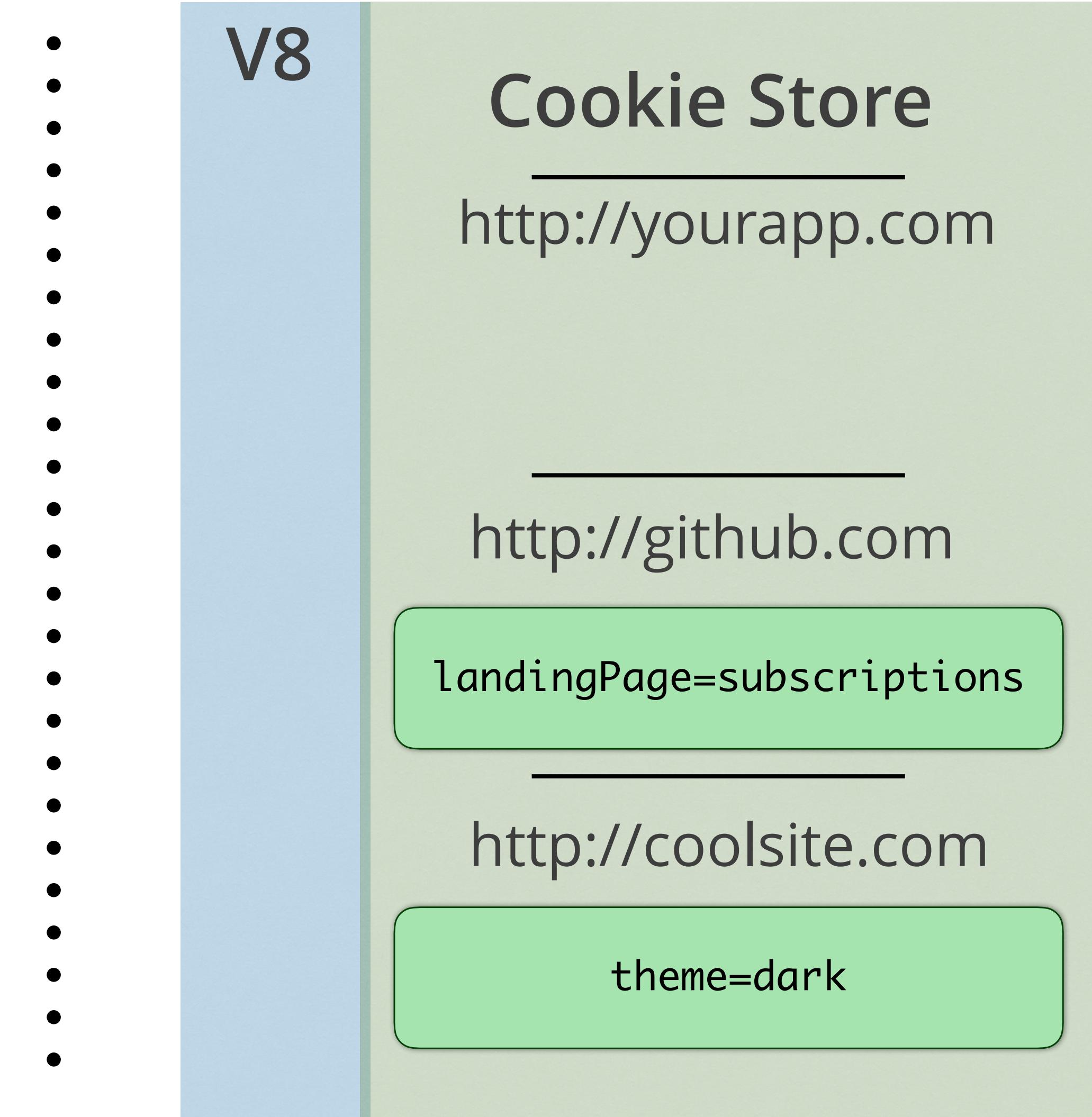
SESSIONS
IN
DETAIL.

Server (Backend)

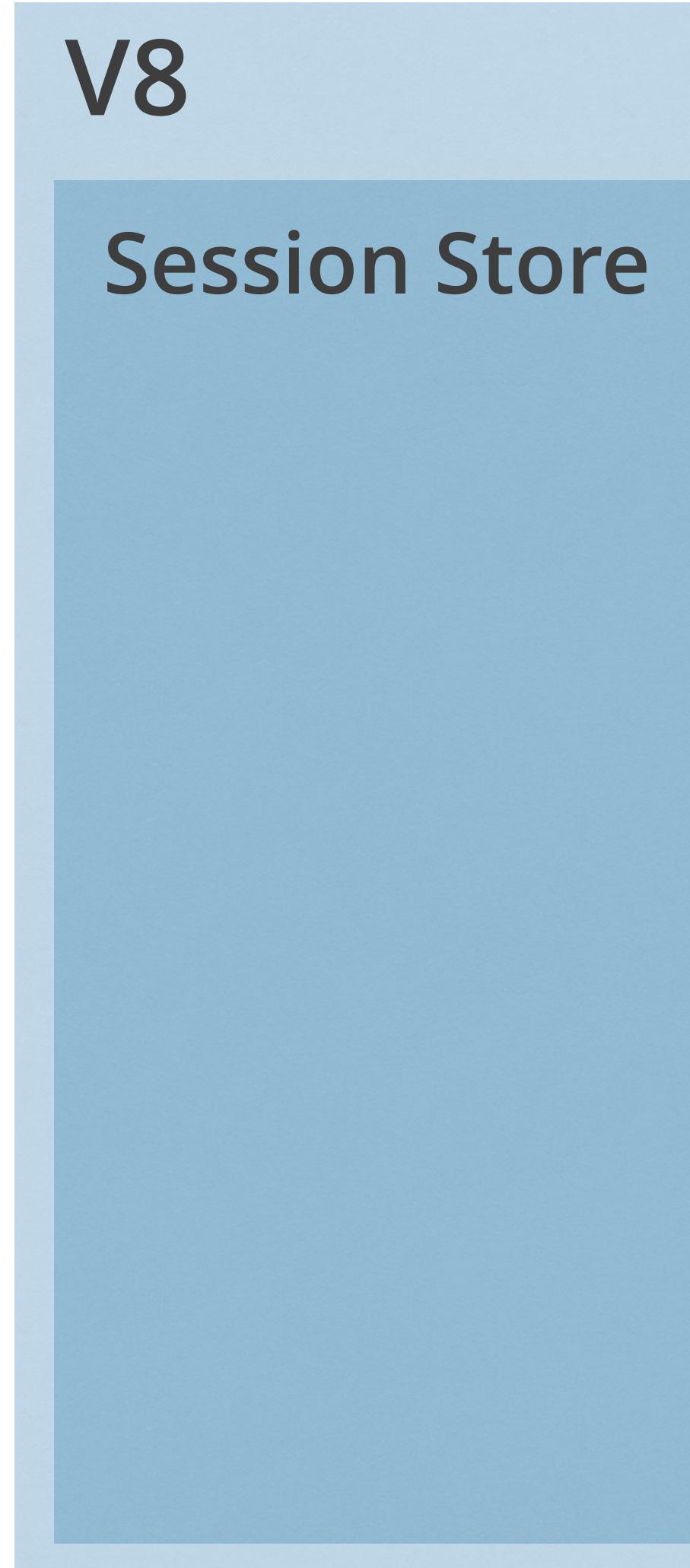


http://yourapp.com

Internet (HTTP)

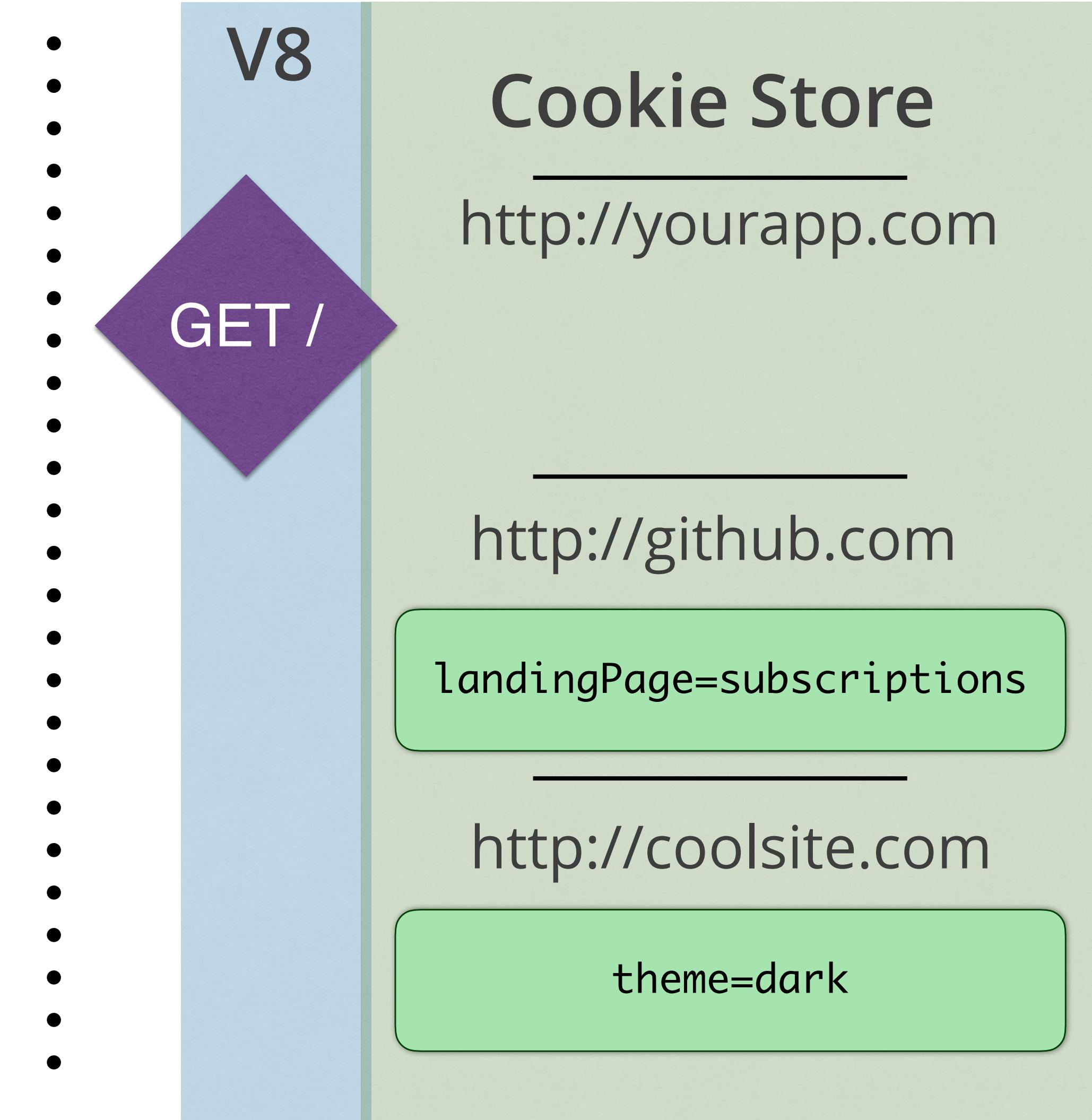


Server (Backend)

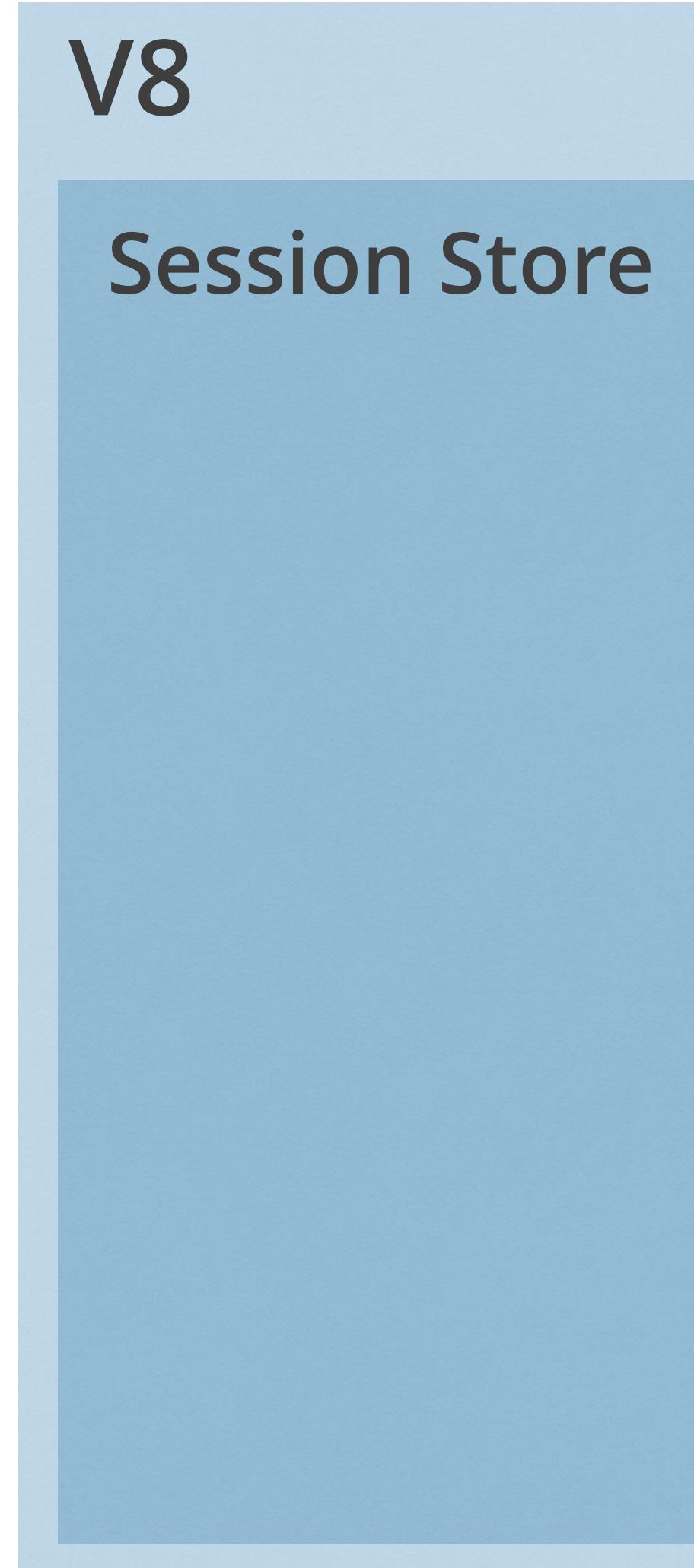


http://yourapp.com

Internet (HTTP)

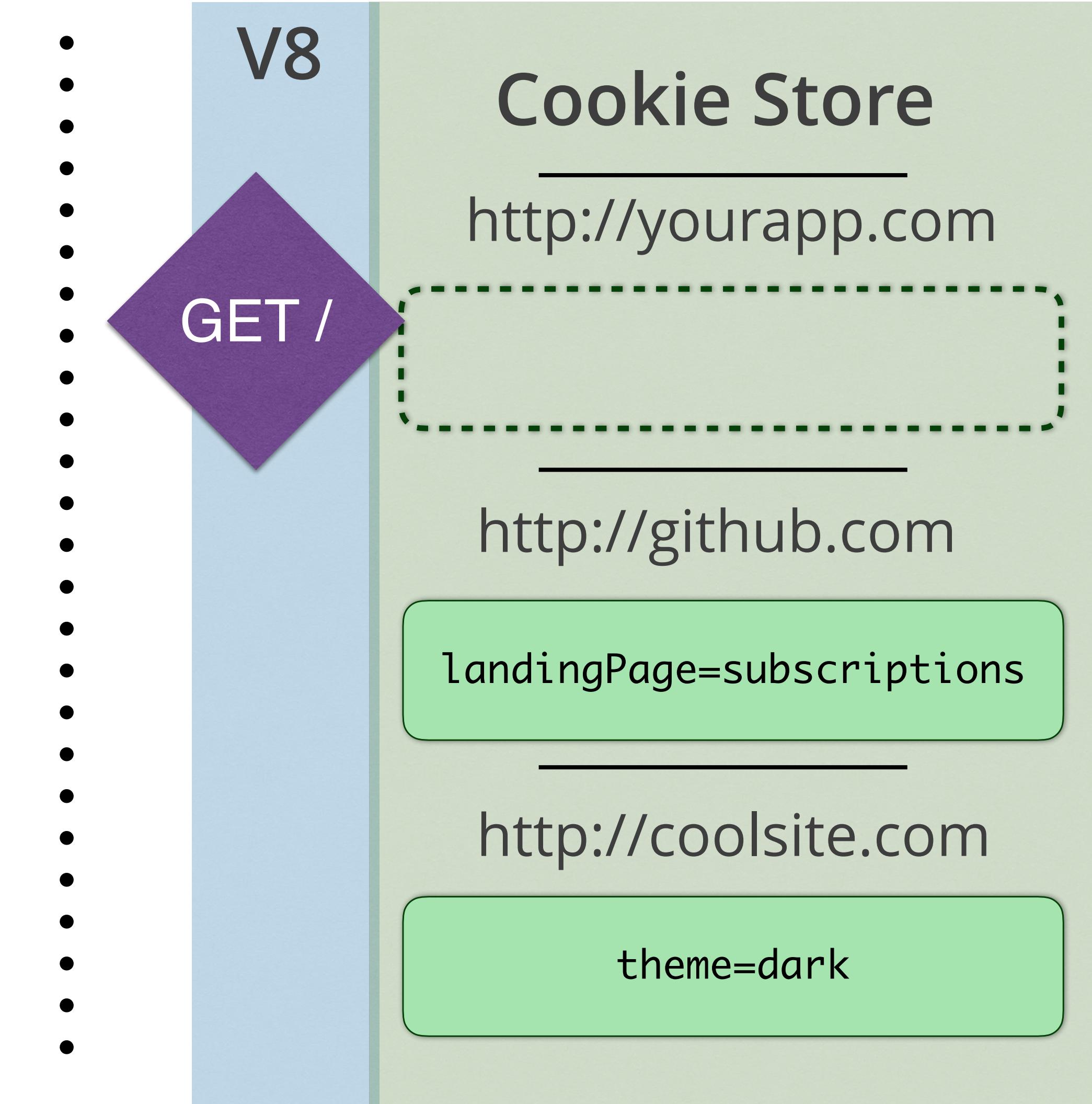


Server (Backend)



<http://yourapp.com>

Internet (HTTP)



HTTP REQUEST

(headers)

(body)

GET / HTTP/1.1
Host: yourapp.com
Connection: keep-alive
Accept: text/html
User-Agent: Chrome/
48.0.2564.116
...etc...

HTTP REQUEST

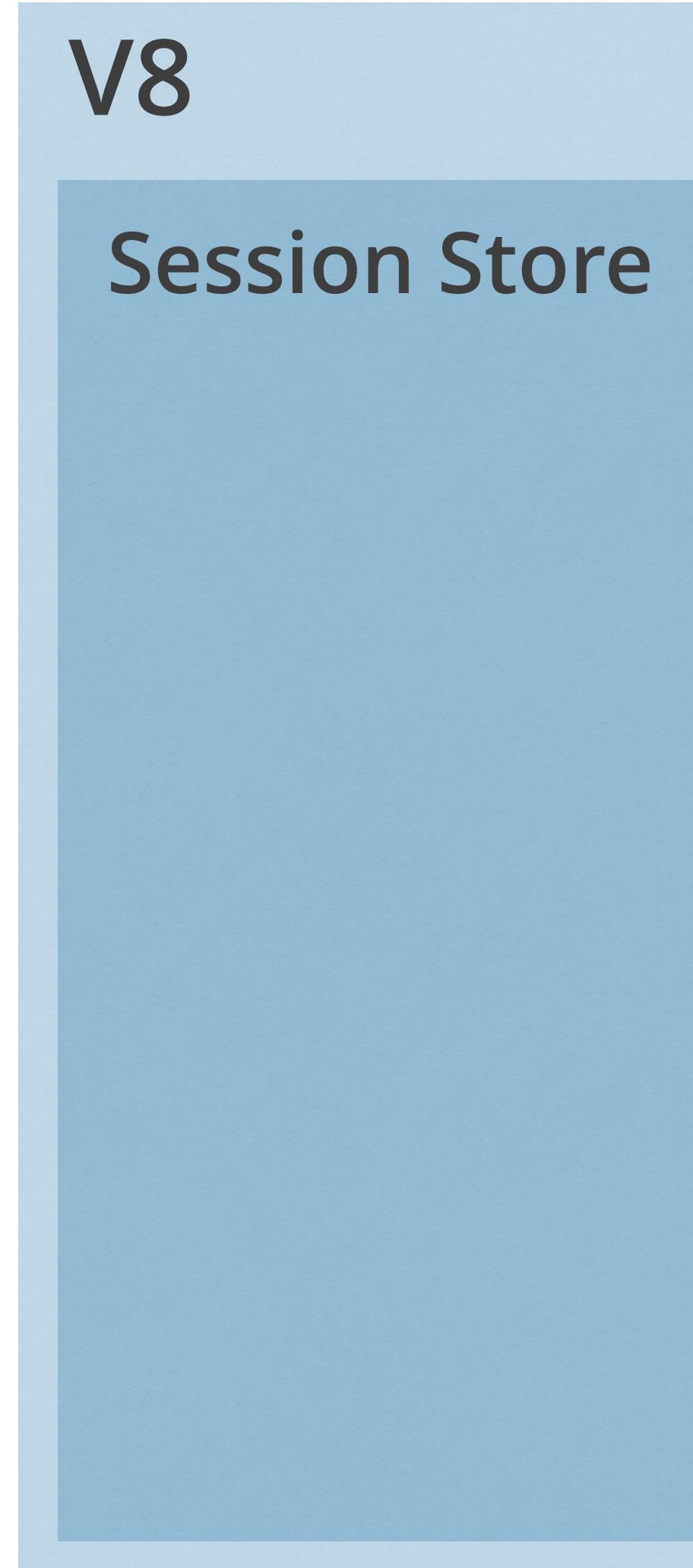
(headers)

(body)

GET / HTTP/1.1
Host: yourapp.com
Connection: keep-alive
Accept: text/html
User-Agent: Chrome/
48.0.2564.116
...etc...

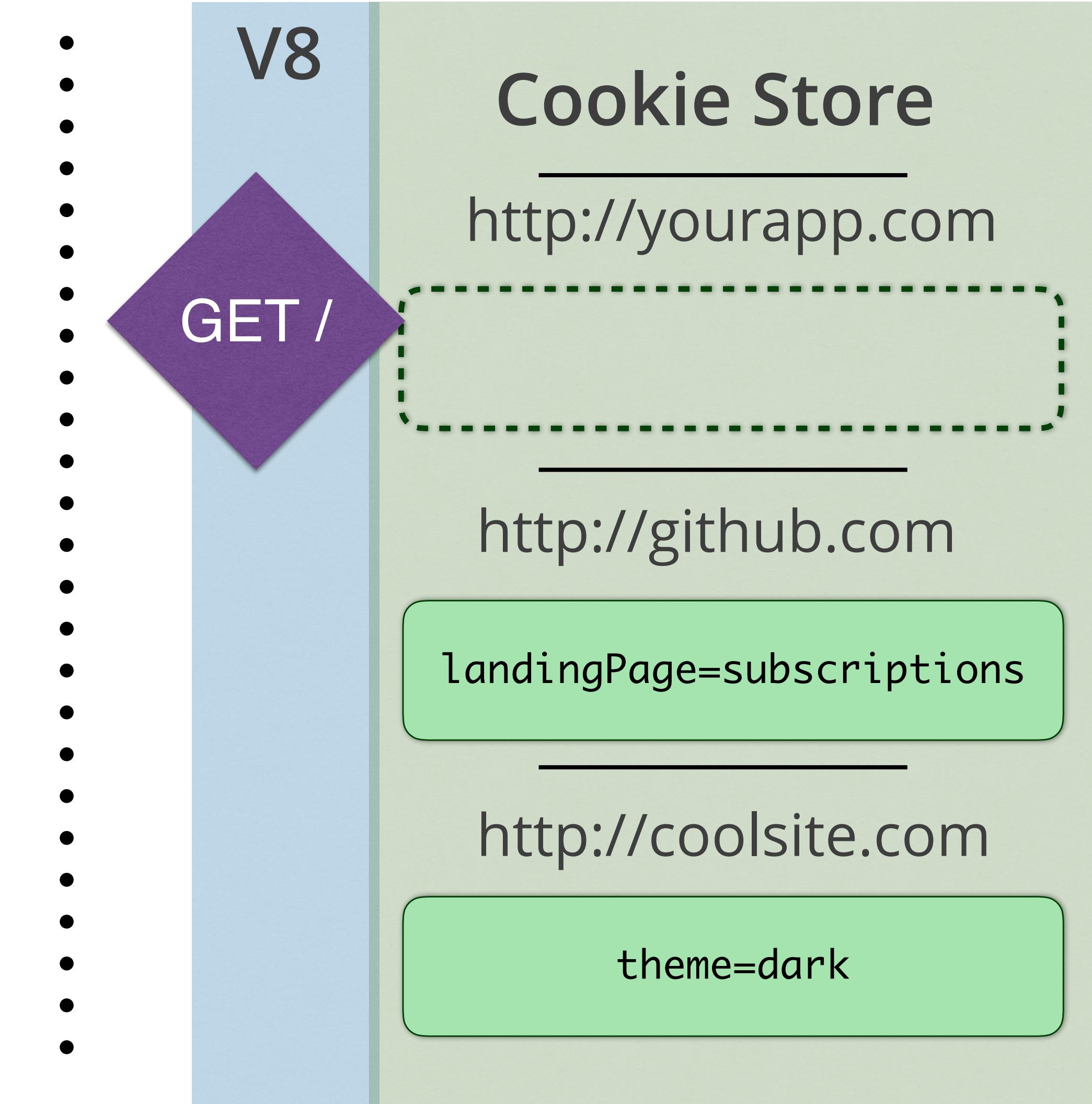
...no cookie info (yet)!

Server (Backend)



<http://yourapp.com>

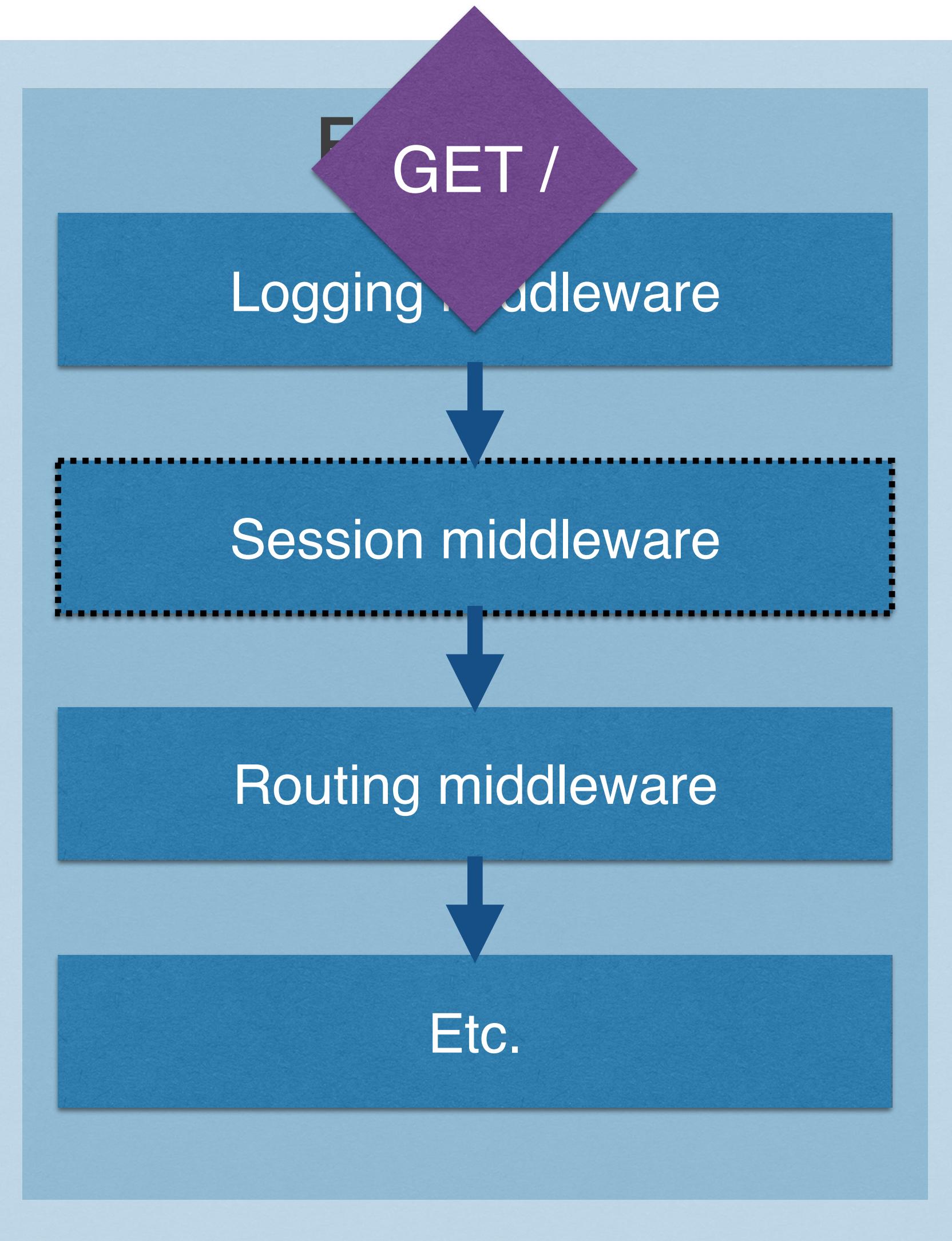
Internet (HTTP)



Server (Backend)

V8

Session Store



<http://yourapp.com>

Internet (HTTP)

V8

Cookie Store

<http://yourapp.com>

A decorative horizontal border at the top of the page. It features a top line made of a dashed green pattern with rounded ends, and a bottom line made of a solid green color.

<http://github.com>

landingPage=subscriptions

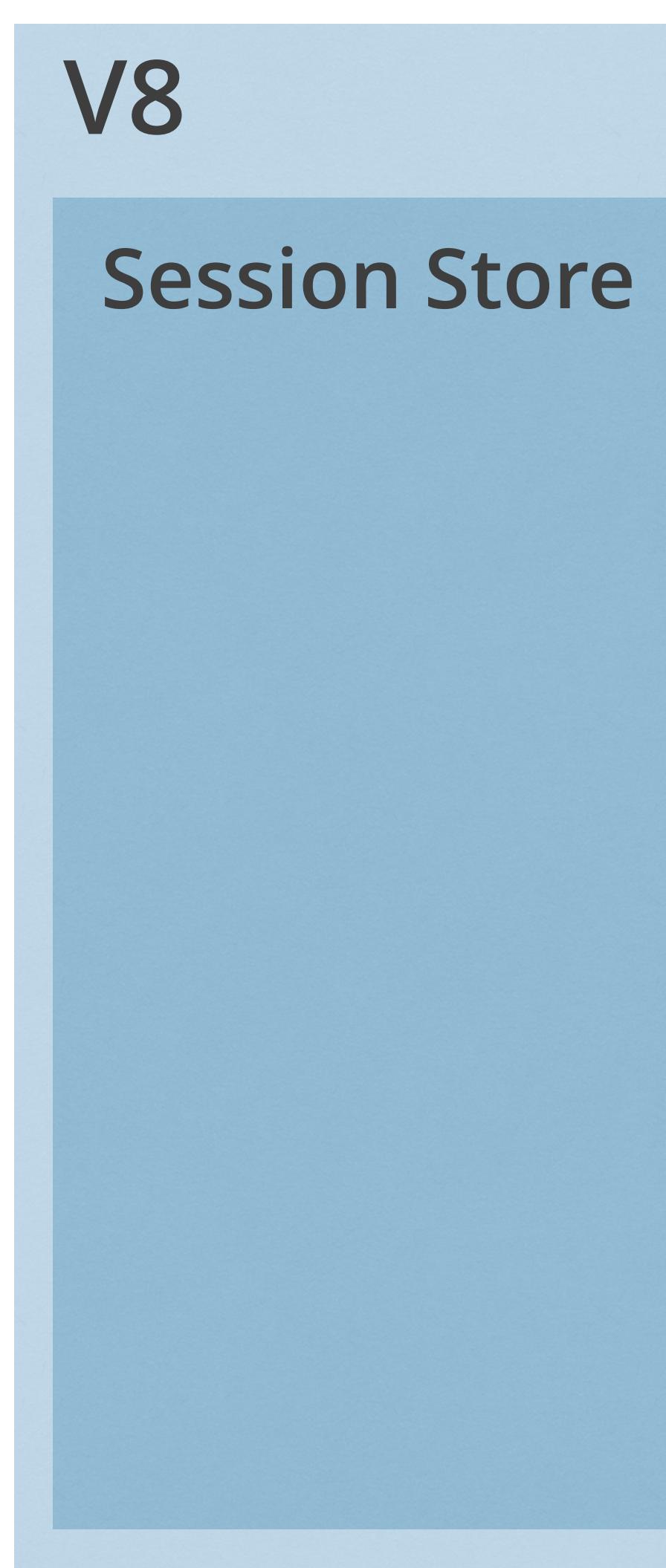
<http://coolsite.com>

theme=dark

Client (Browser)

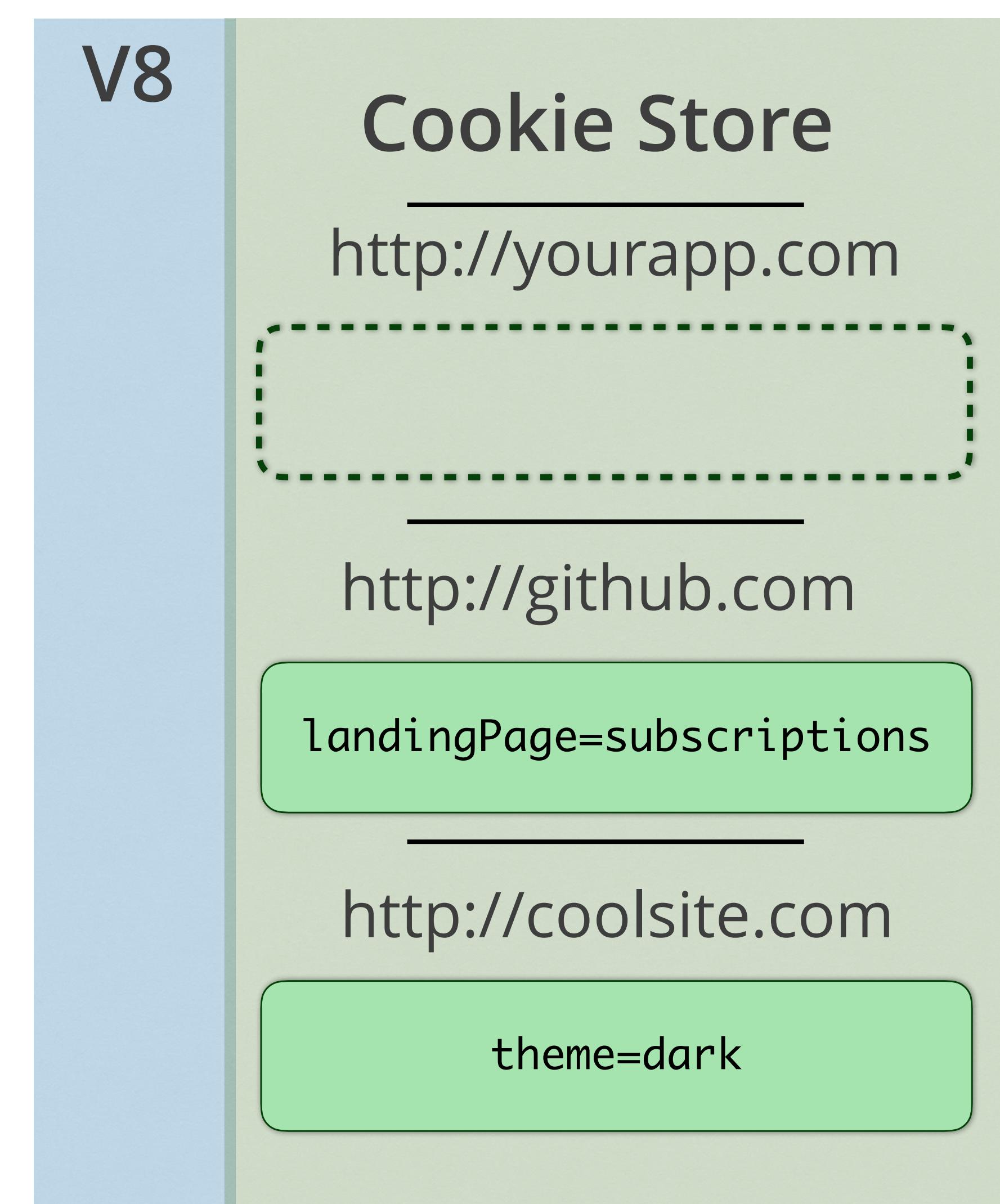
AUTH

Server (Backend)

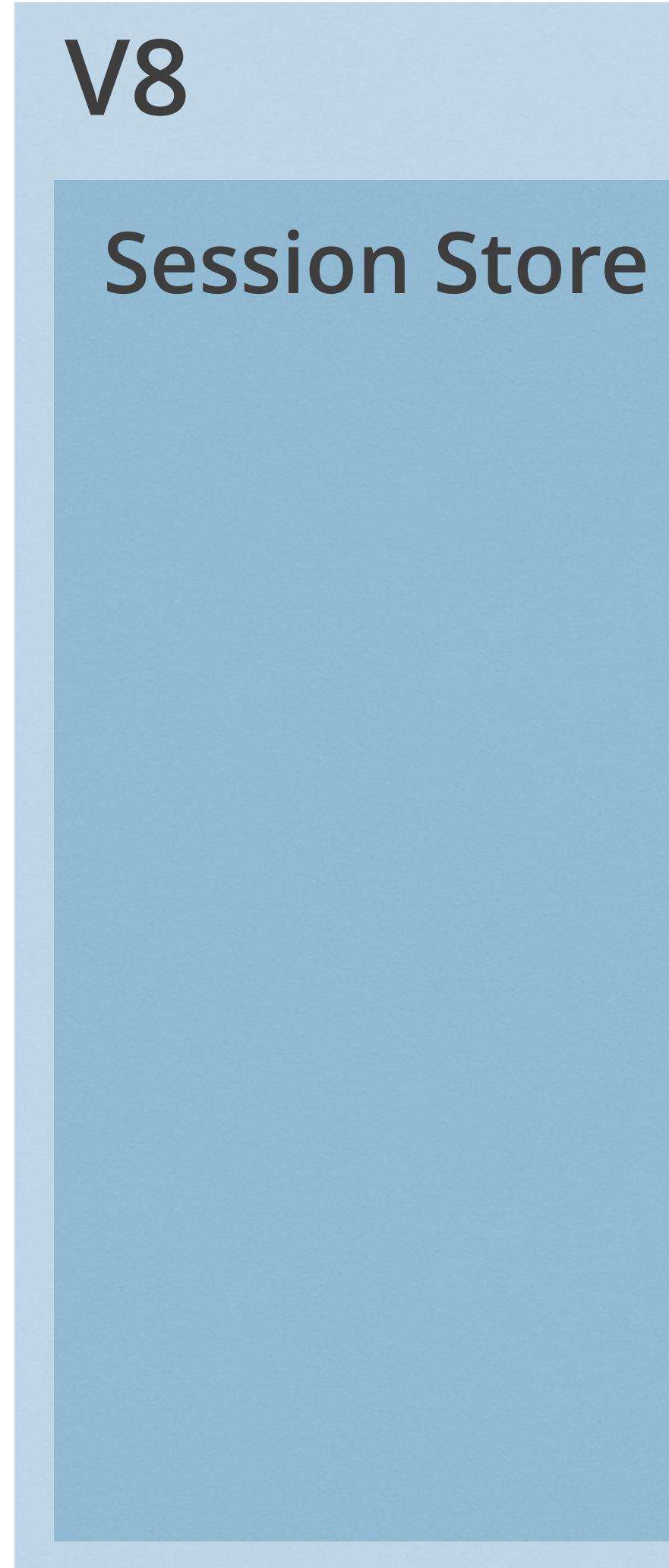


<http://yourapp.com>

Internet (HTTP)

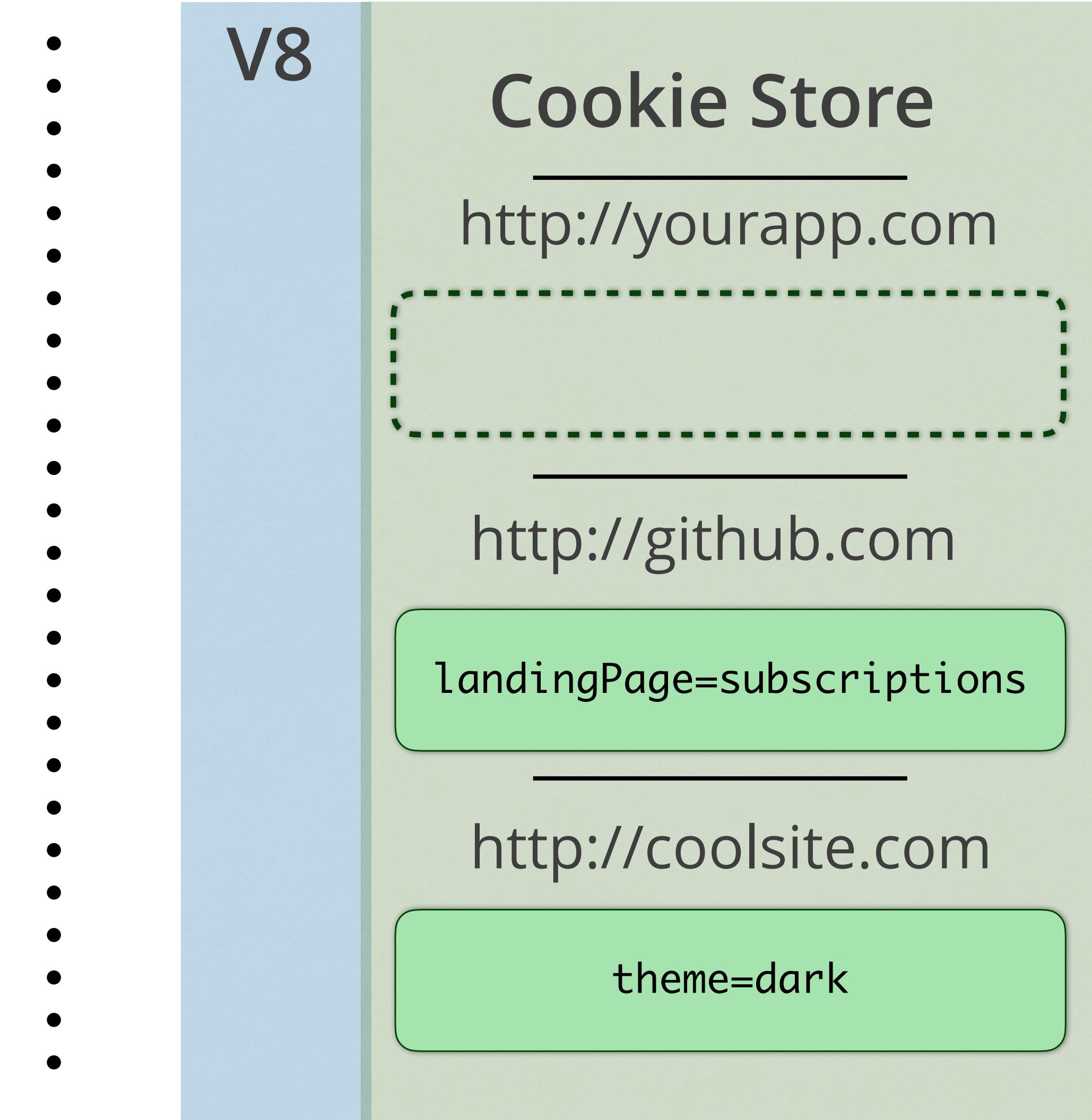


Server (Backend)



<http://yourapp.com>

Internet (HTTP)



HTTP REQUEST

(headers)

(body)

GET / HTTP/1.1
Host: yourapp.com
Connection: keep-alive
Accept: text/html
User-Agent: Chrome/
48.0.2564.116
...etc...

HTTP REQUEST

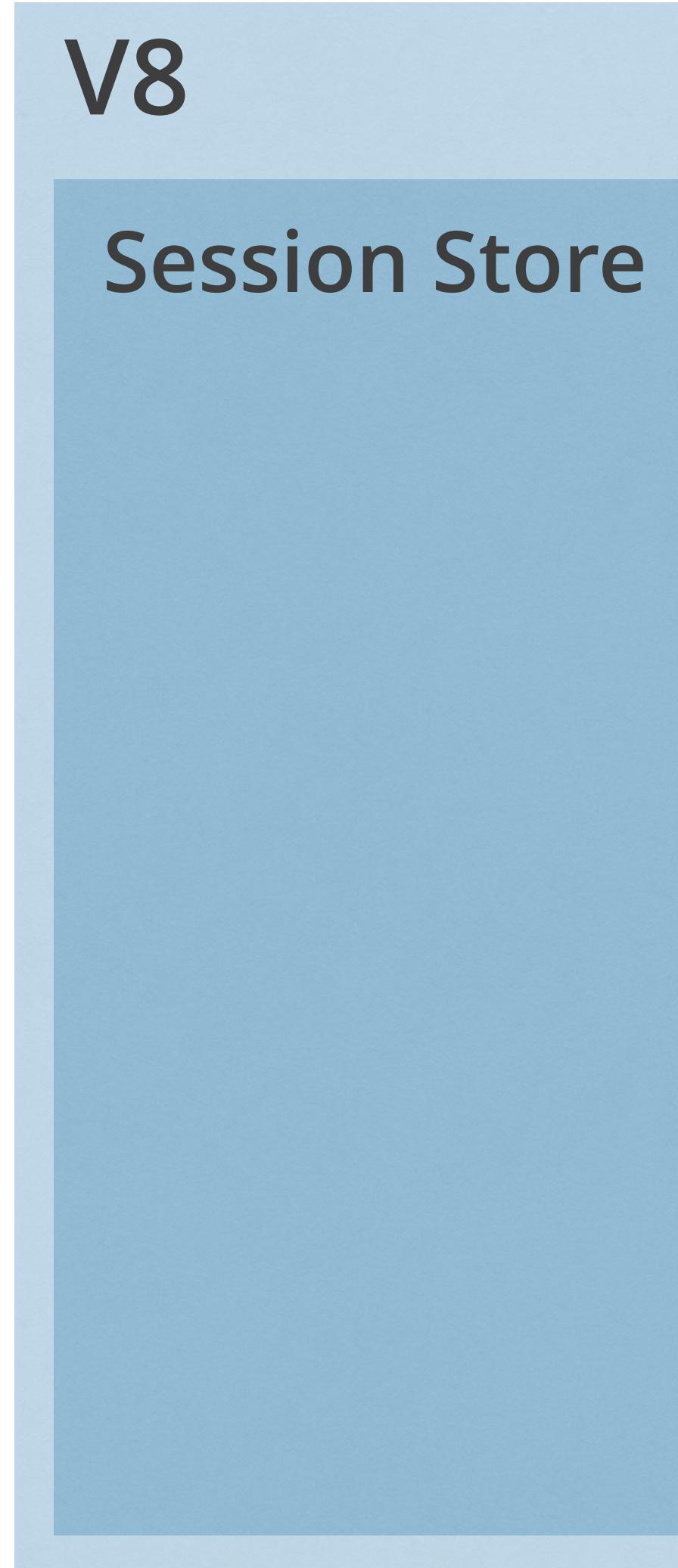
(headers)

(body)

GET / HTTP/1.1
Host: yourapp.com
Connection: keep-alive
Accept: text/html
User-Agent: Chrome/
48.0.2564.116
...etc...

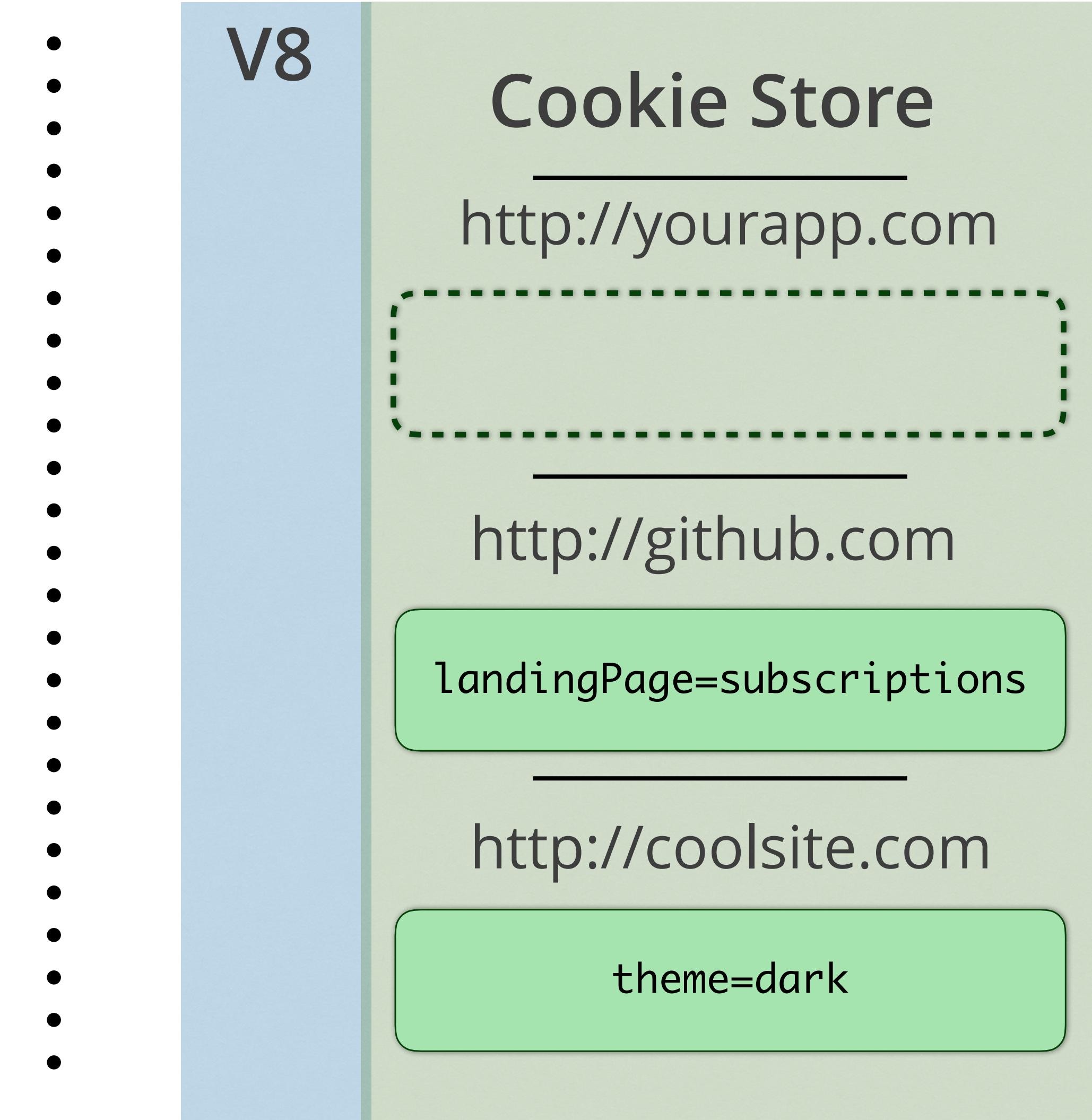
still no cookie info...
session middleware:
"let's make a session!"

Server (Backend)

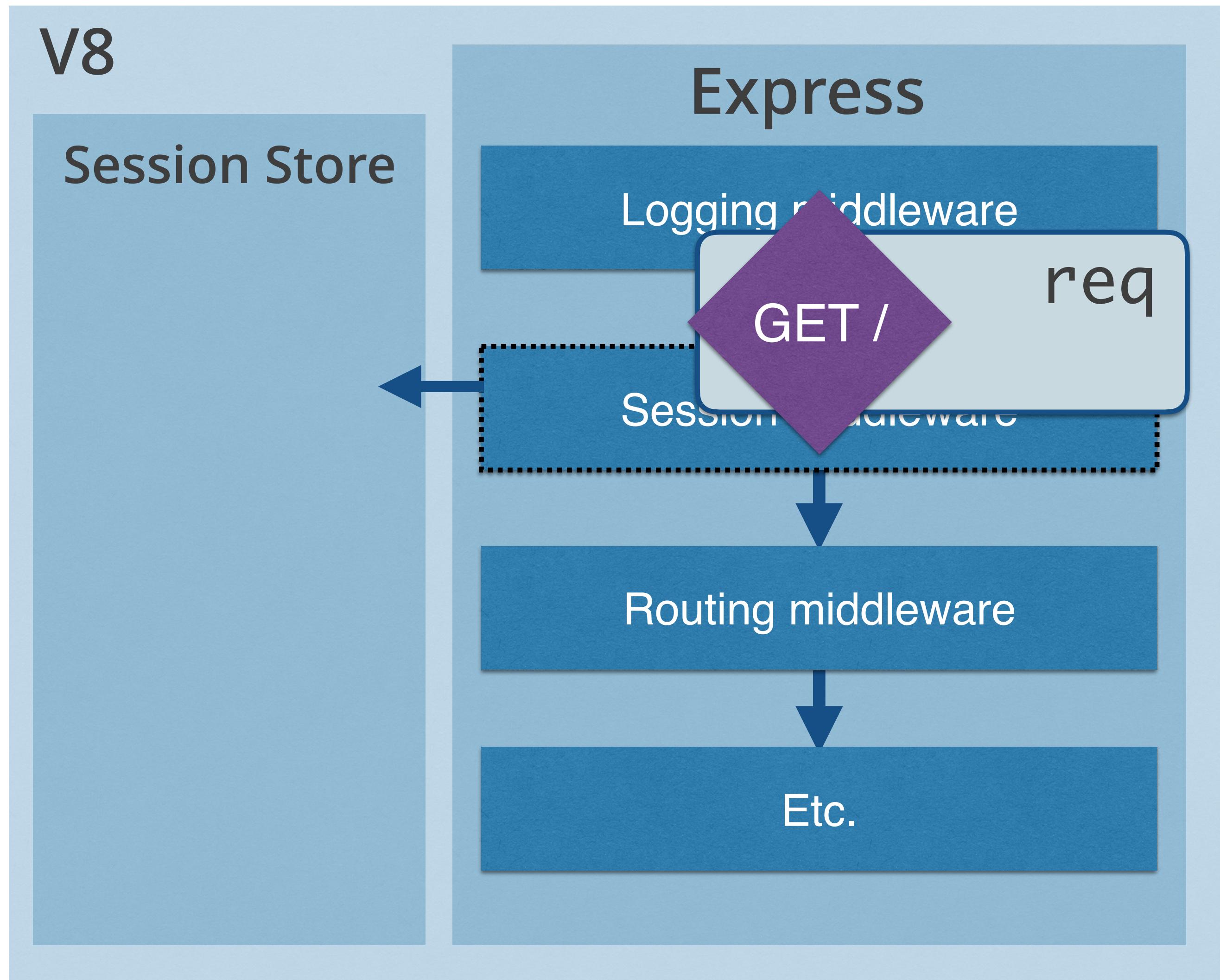


<http://yourapp.com>

Internet (HTTP)

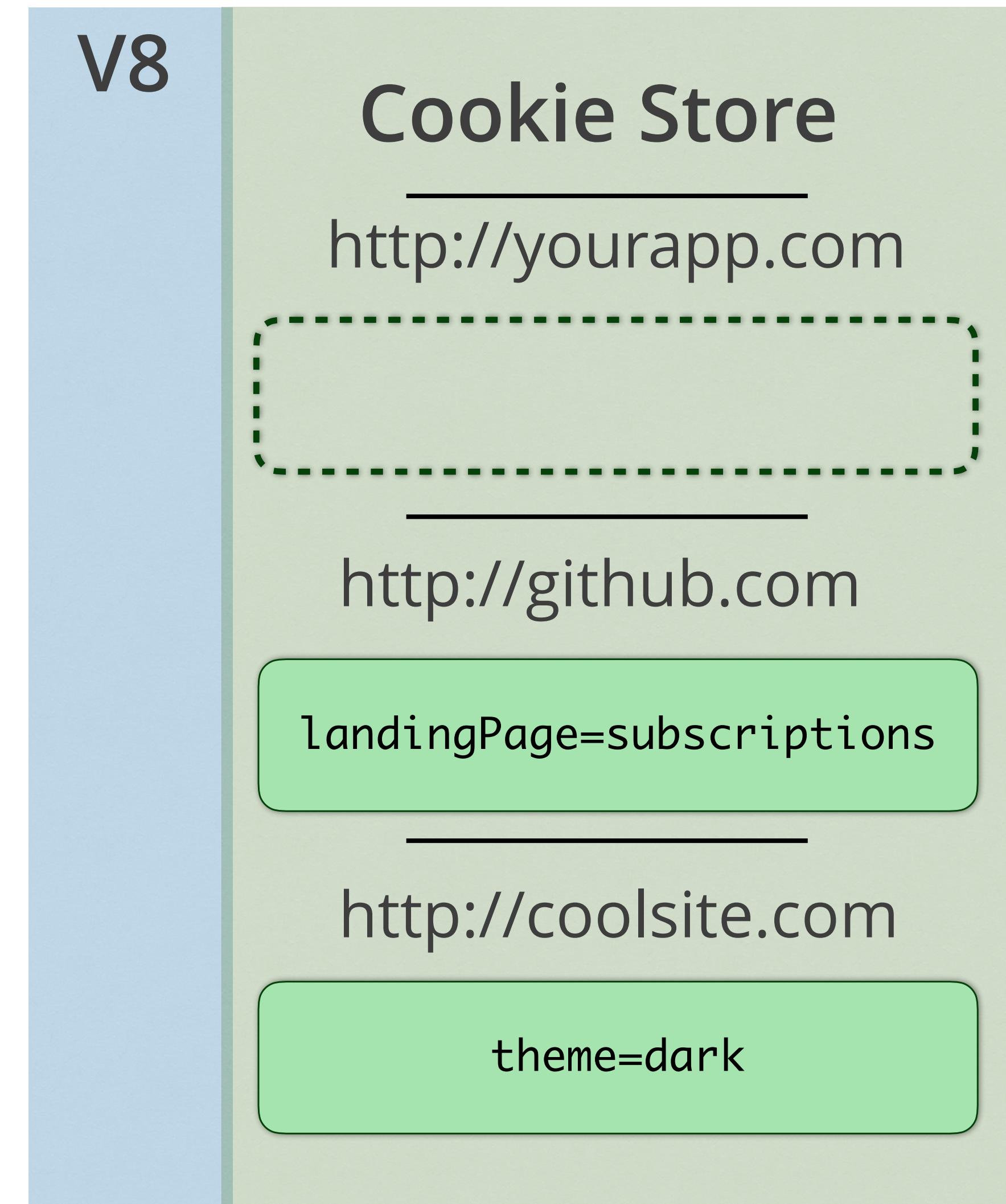


Server (Backend)

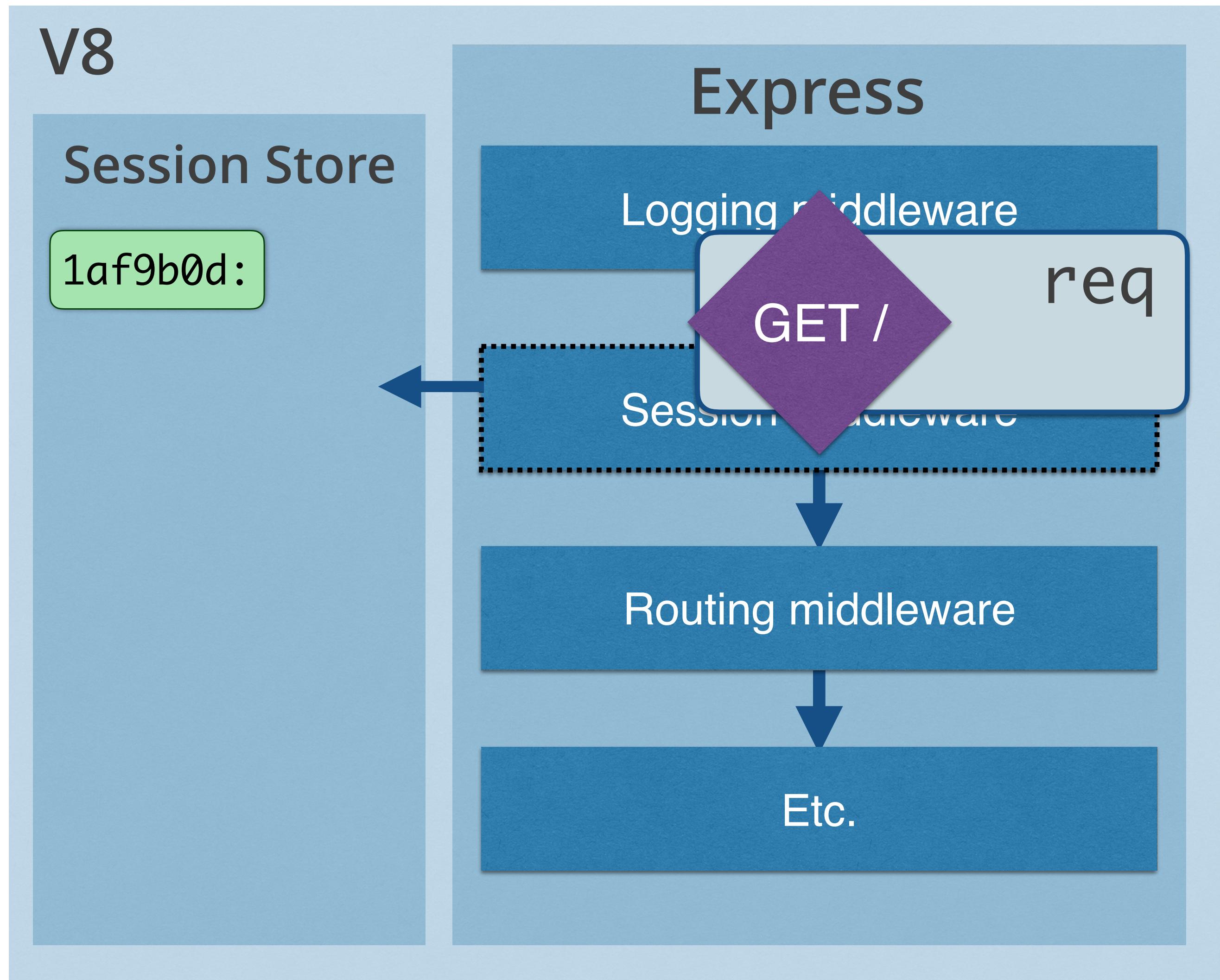


<http://yourapp.com>

Internet (HTTP)

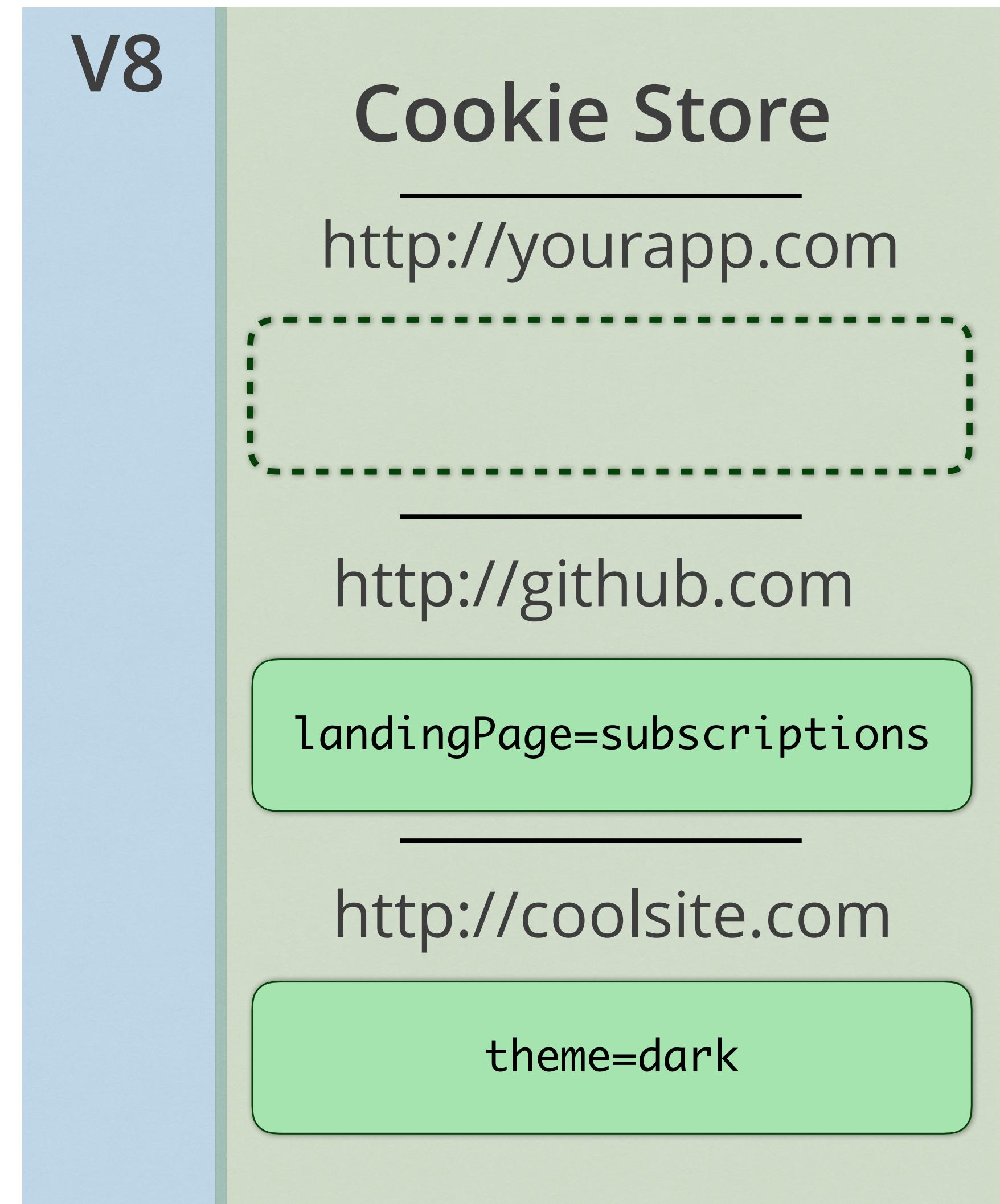


Server (Backend)

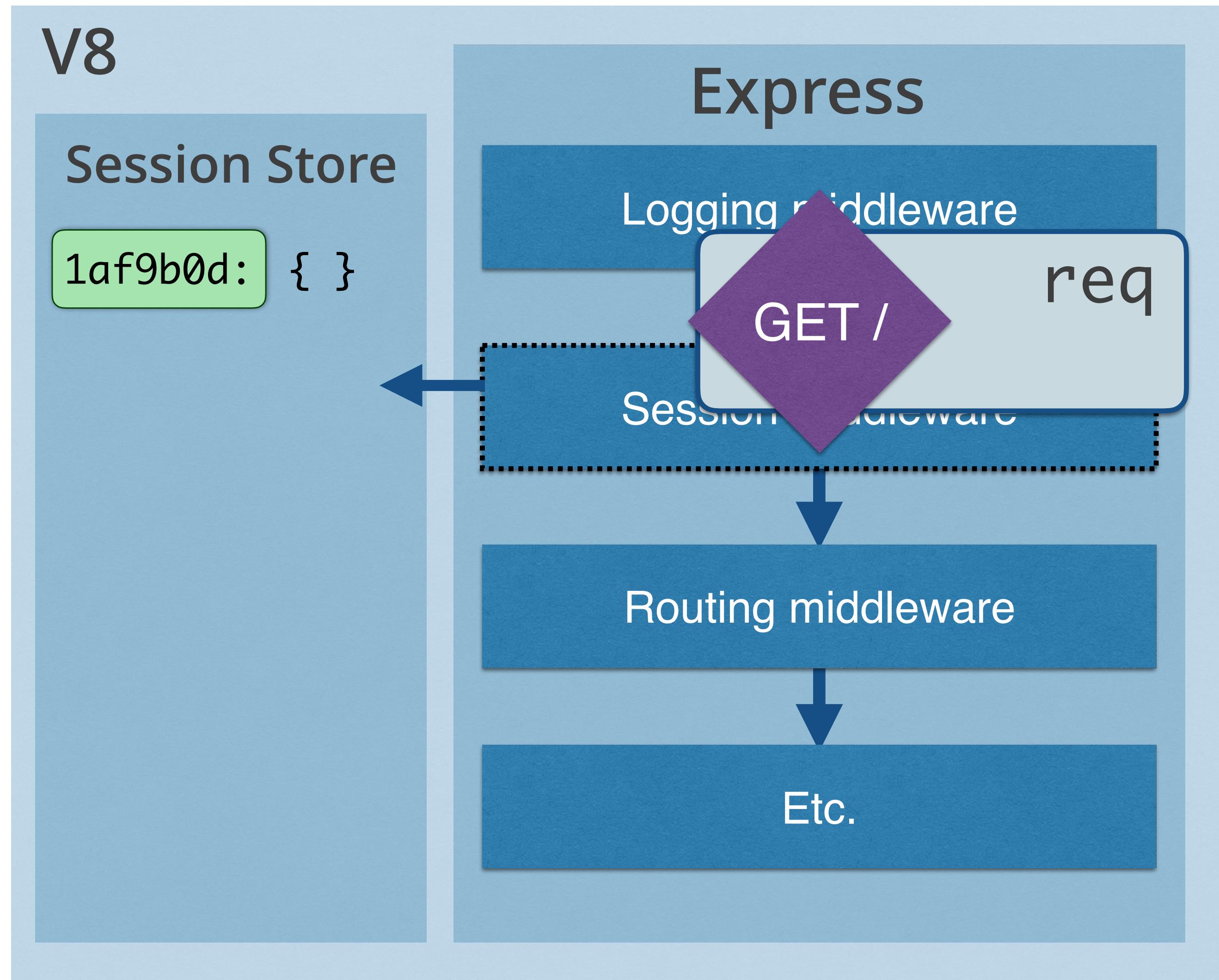


<http://yourapp.com>

Internet (HTTP)

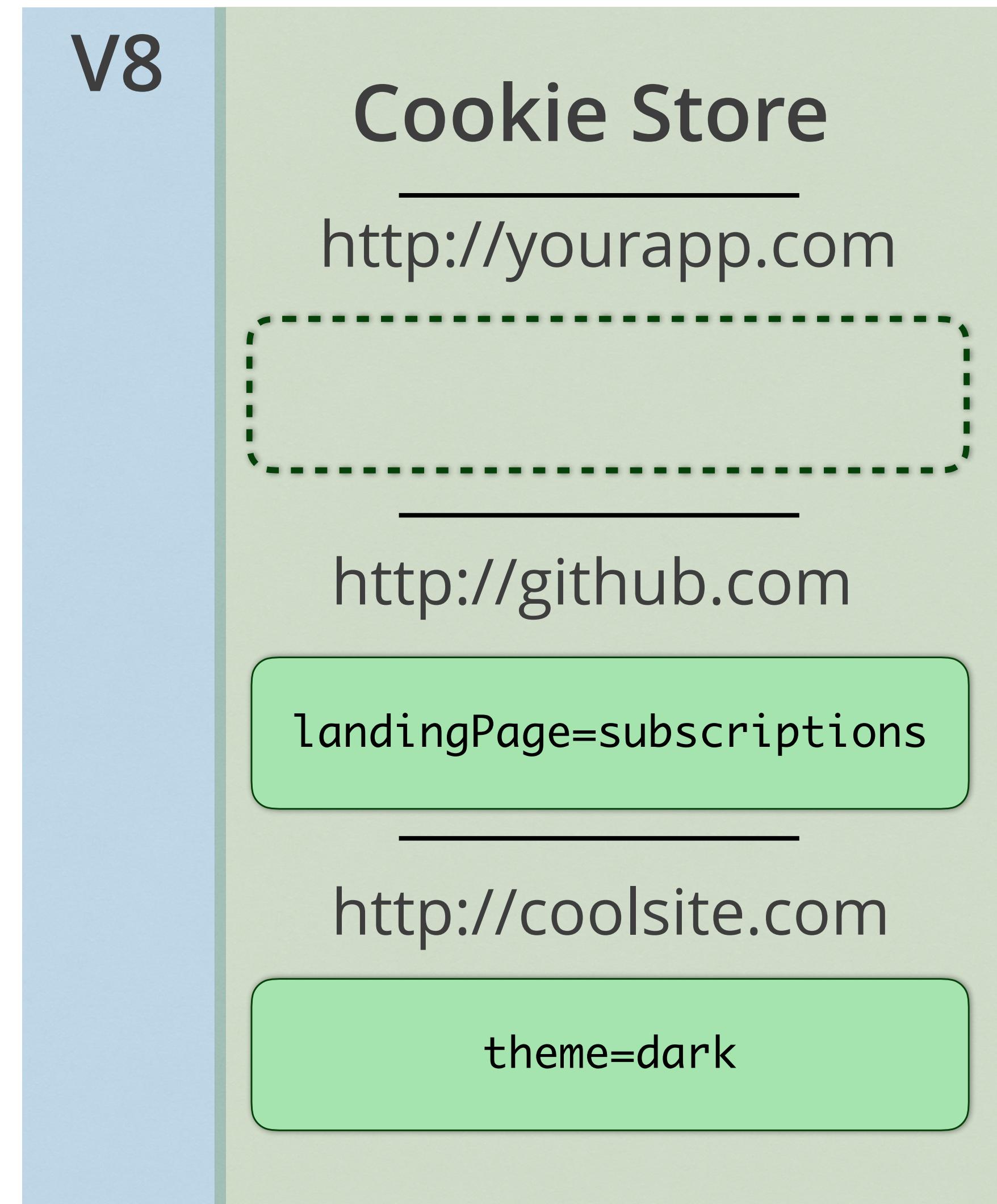


Server (Backend)

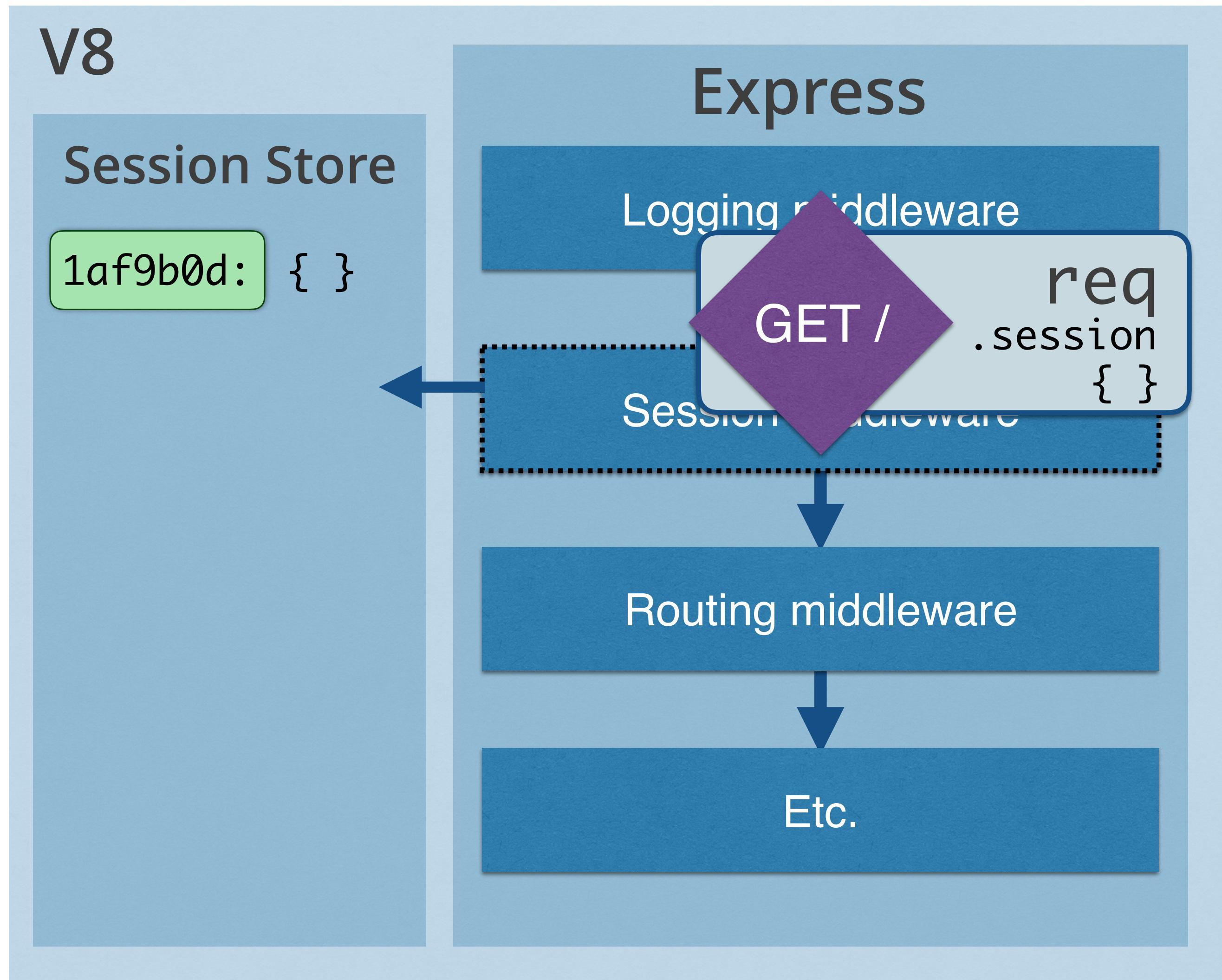


<http://yourapp.com>

Internet (HTTP)

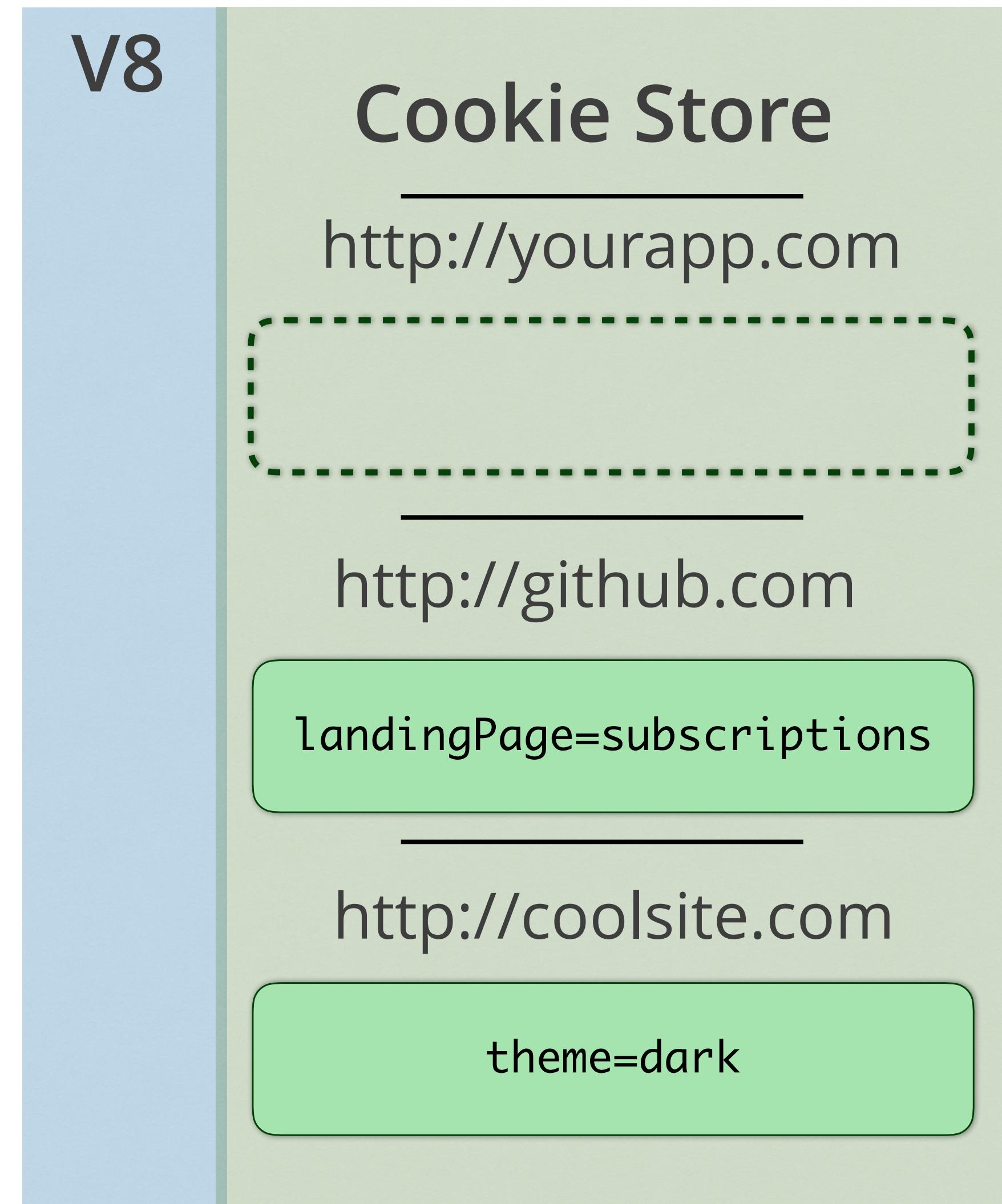


Server (Backend)

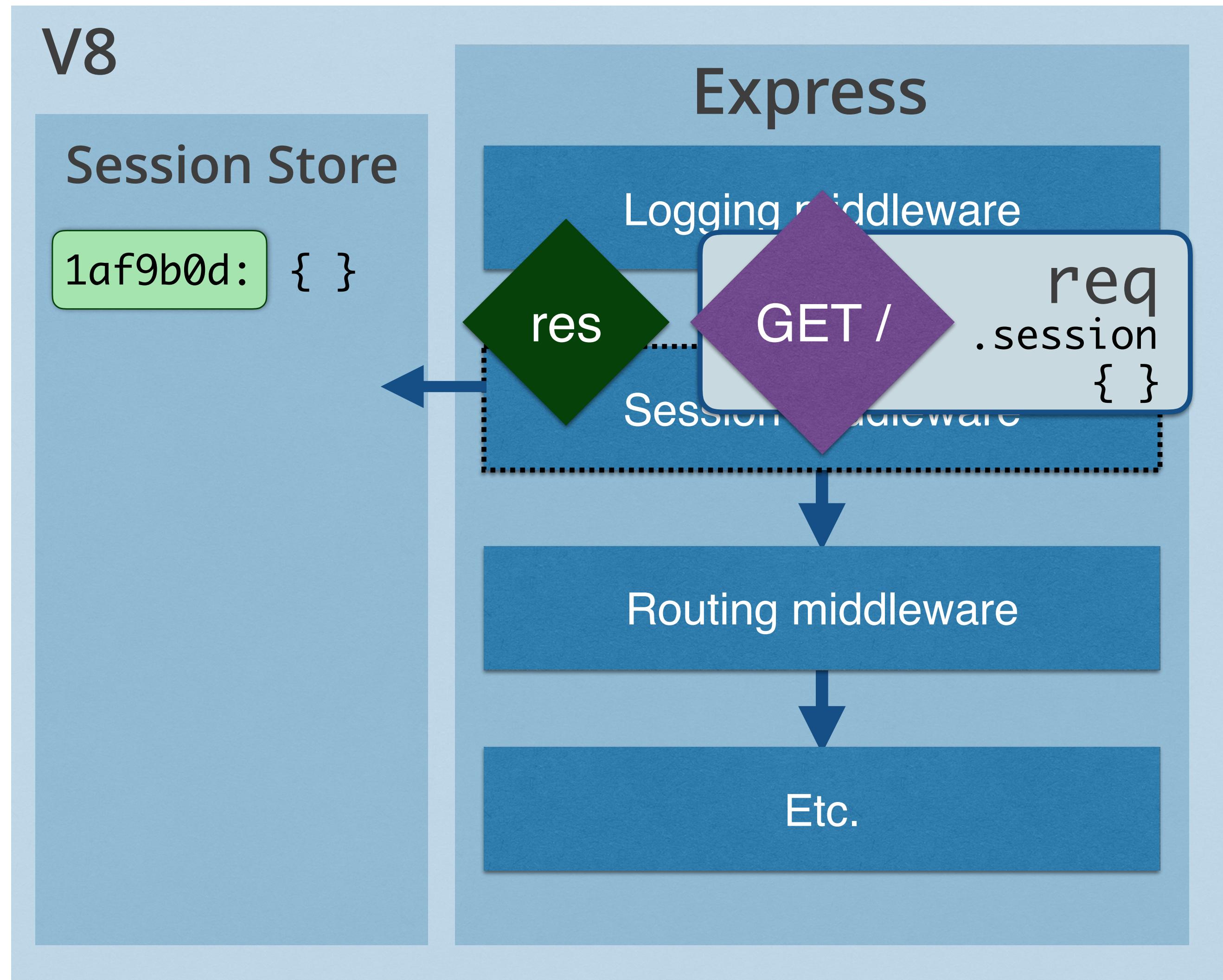


<http://yourapp.com>

Internet (HTTP)

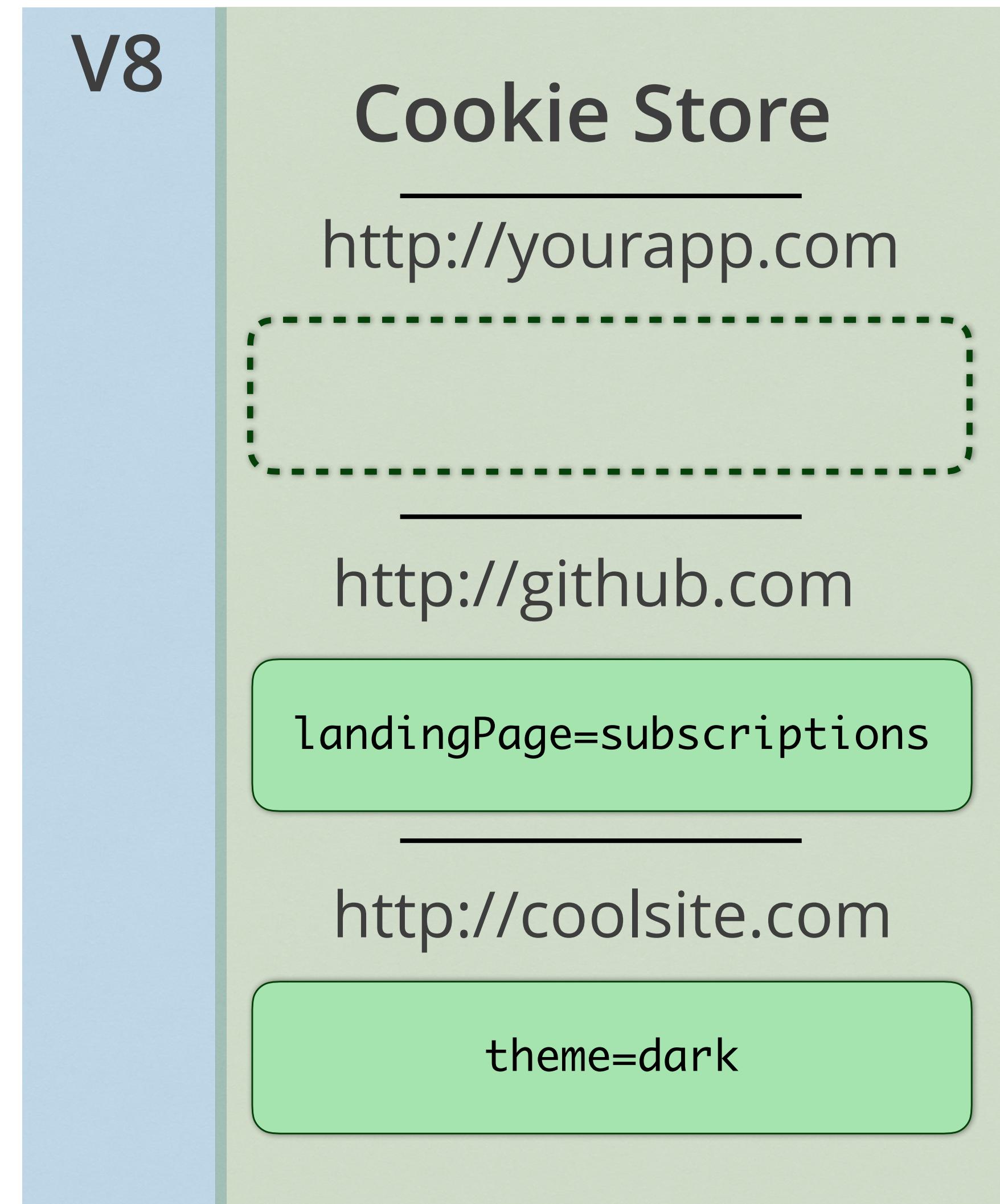


Server (Backend)



<http://yourapp.com>

Internet (HTTP)



Client (Browser)

HTTP RESPONSE

(headers)

(body, after
routing is
done)

HTTP/1.x 200 OK
Transfer-Encoding: chunked
Date: Mon, 22 Feb 2016 18:30:00 GMT
Content-Type: text/html
Content-Encoding: gzip

```
<!DOCTYPE html>
<html>
  <head>
    <title>Your Sweet App</title>
    <script src="/js/main.js"></script>
  </head> ...etc.
```

HTTP RESPONSE

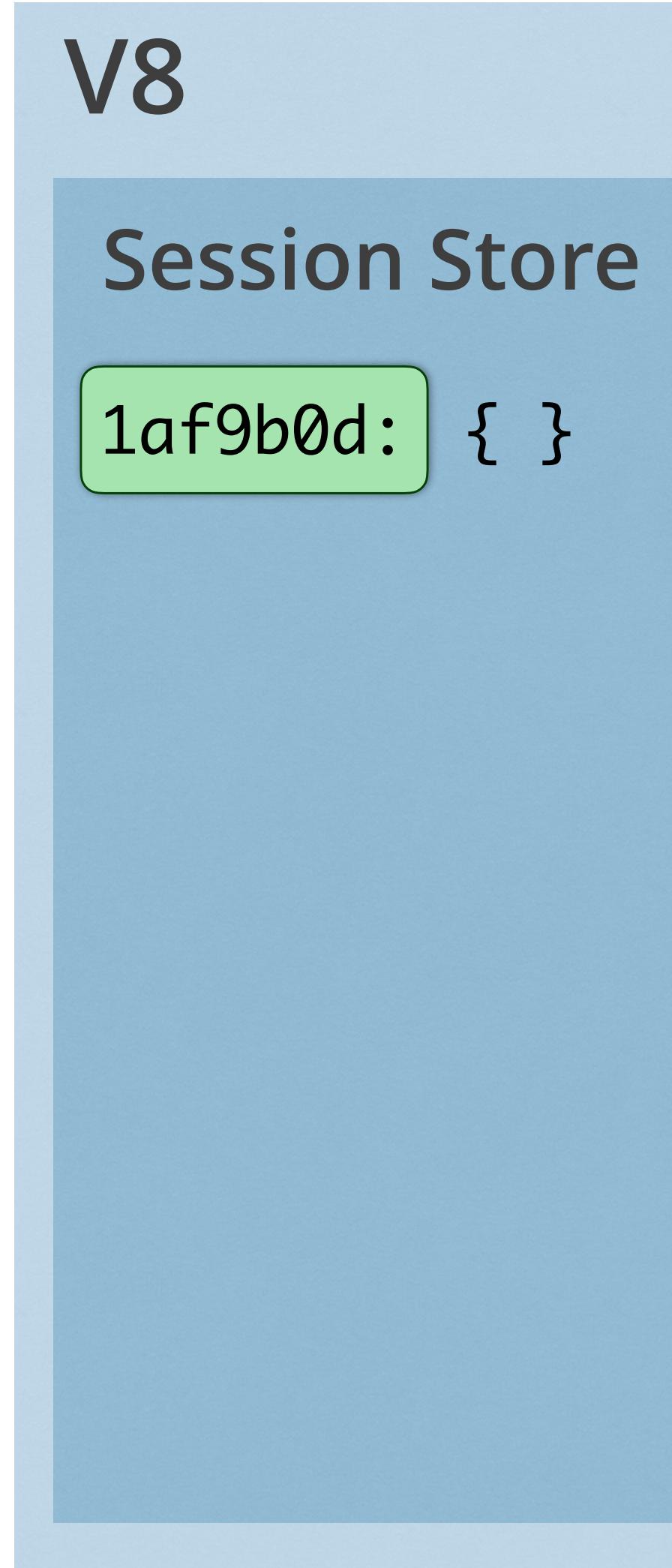
(headers)

HTTP/1.x 200 OK
Transfer-Encoding: chunked
Date: Mon, 22 Feb 2016 18:30:00 GMT
Content-Type: text/html
Content-Encoding: gzip
set-cookie: myUid=1af9b0d

(body, after
routing is
done)

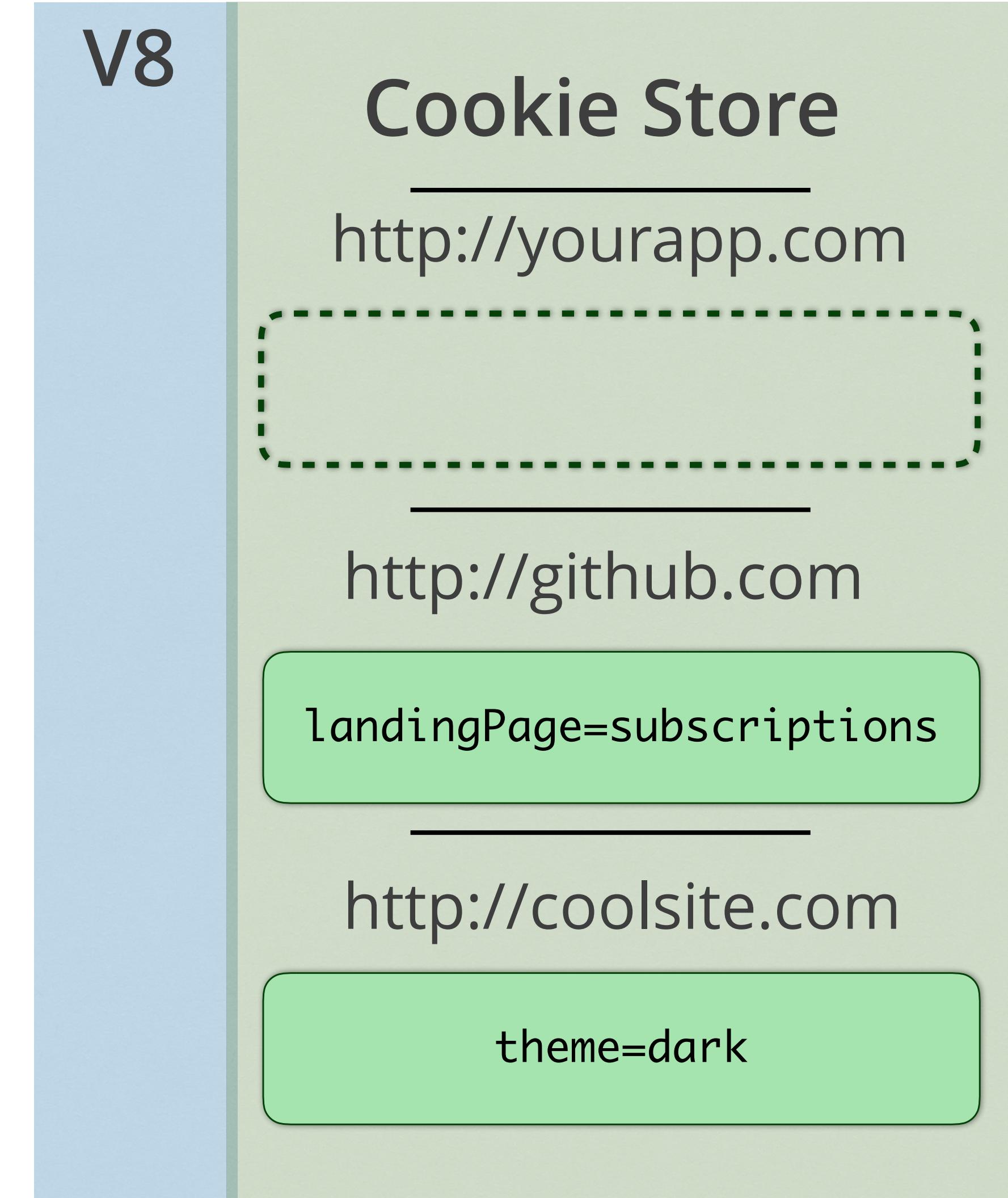
```
<!DOCTYPE html>
<html>
  <head>
    <title>Your Sweet App</title>
    <script src="/js/main.js"></script>
  </head> ...etc.
```

Server (Backend)

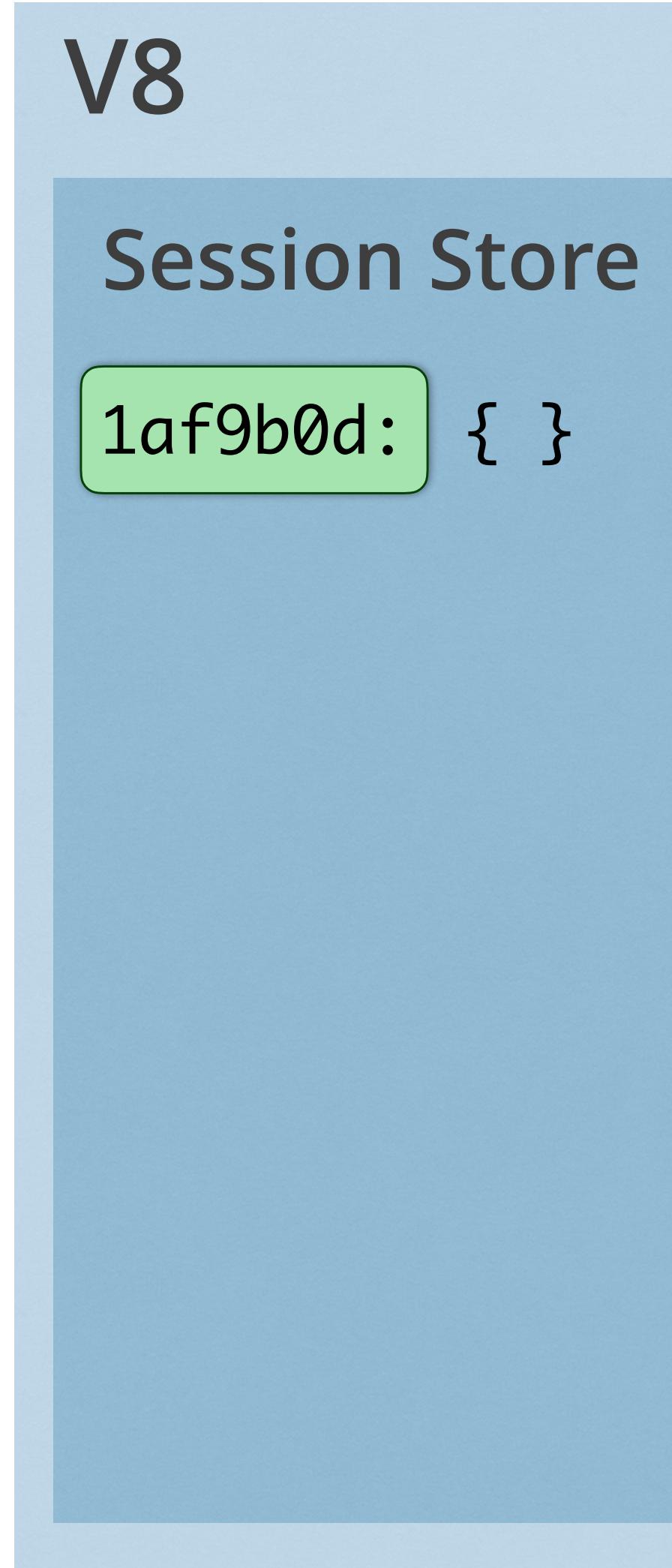


<http://yourapp.com>

Internet (HTTP)

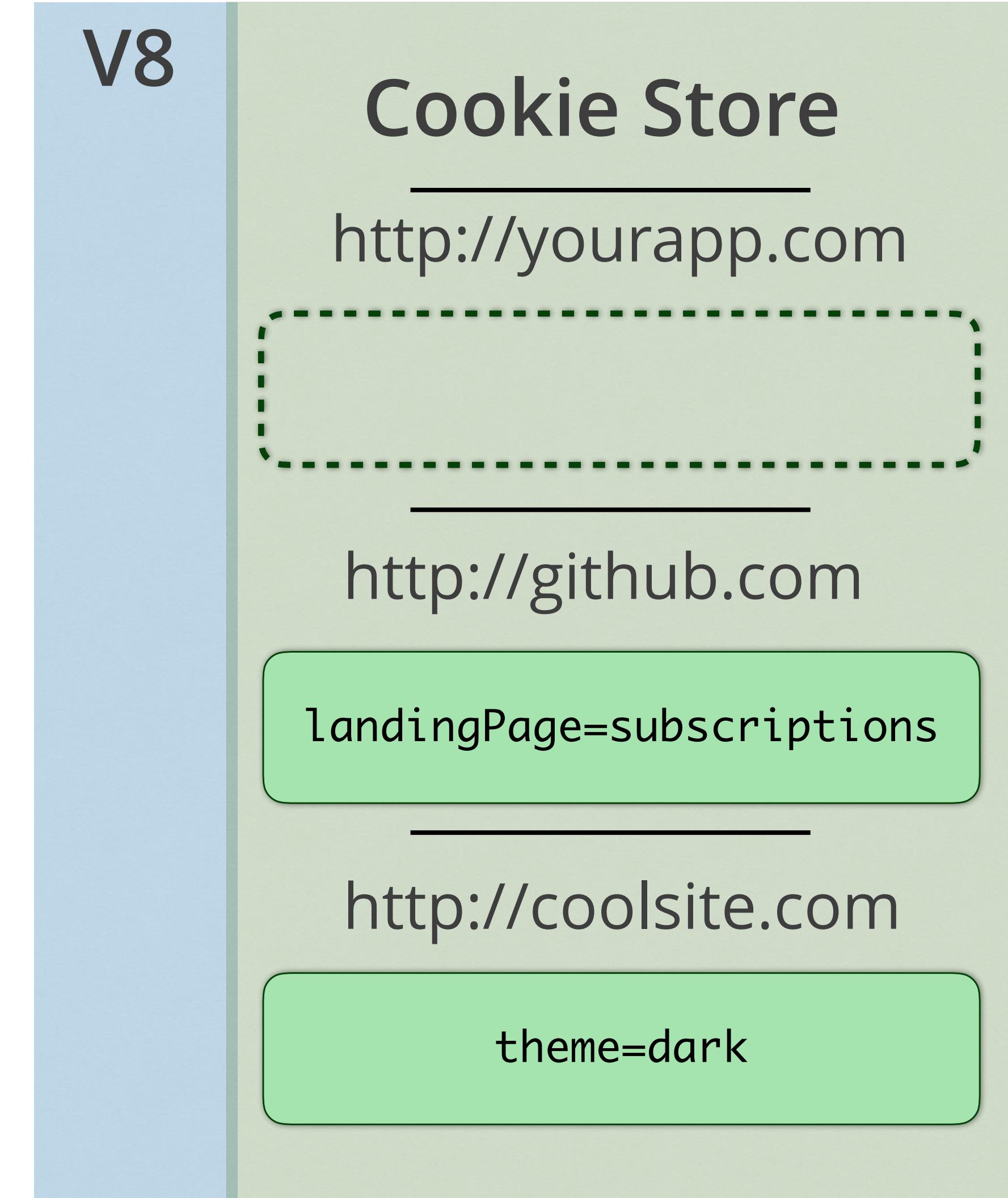


Server (Backend)



<http://yourapp.com>

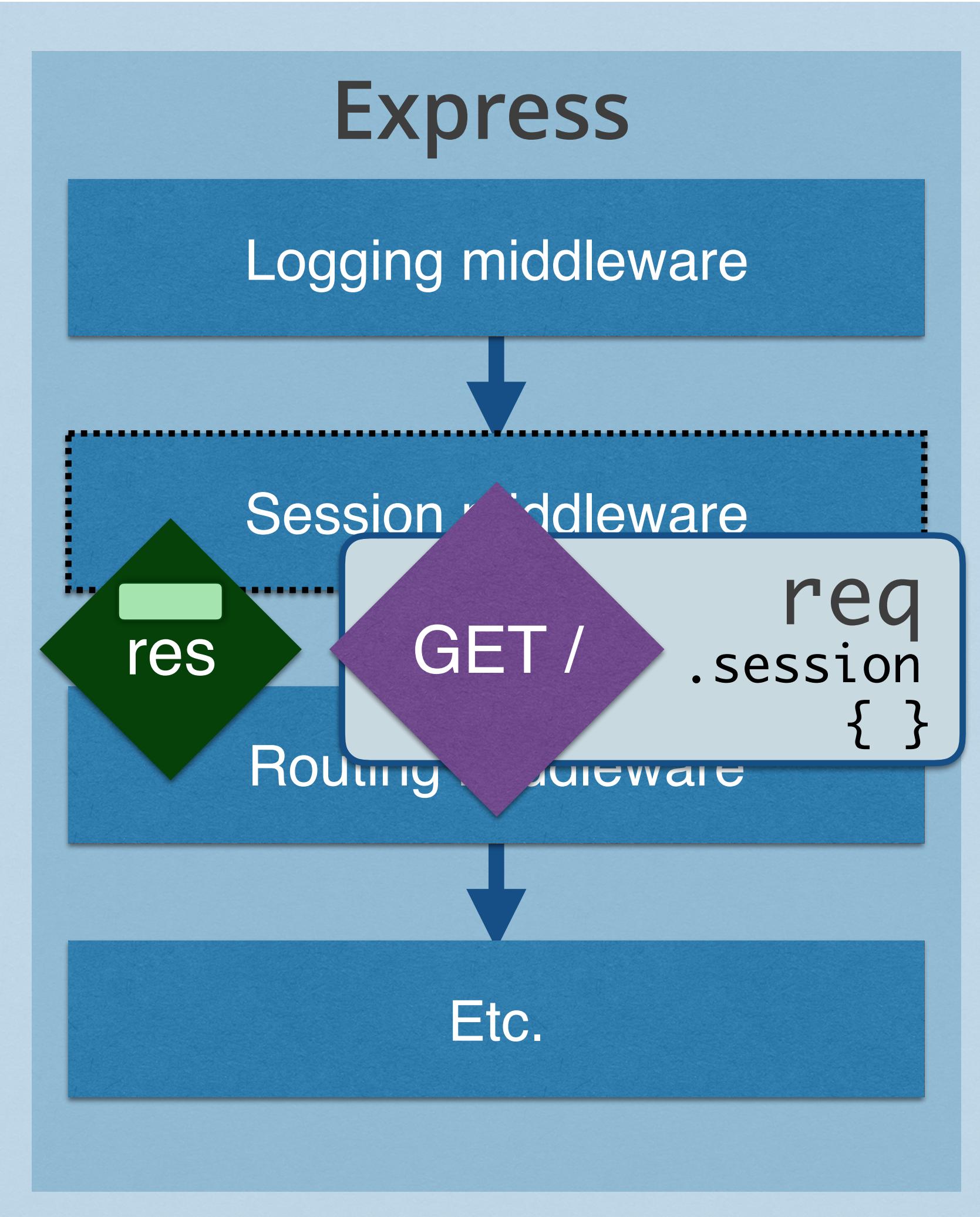
Internet (HTTP)



Server (Backend)

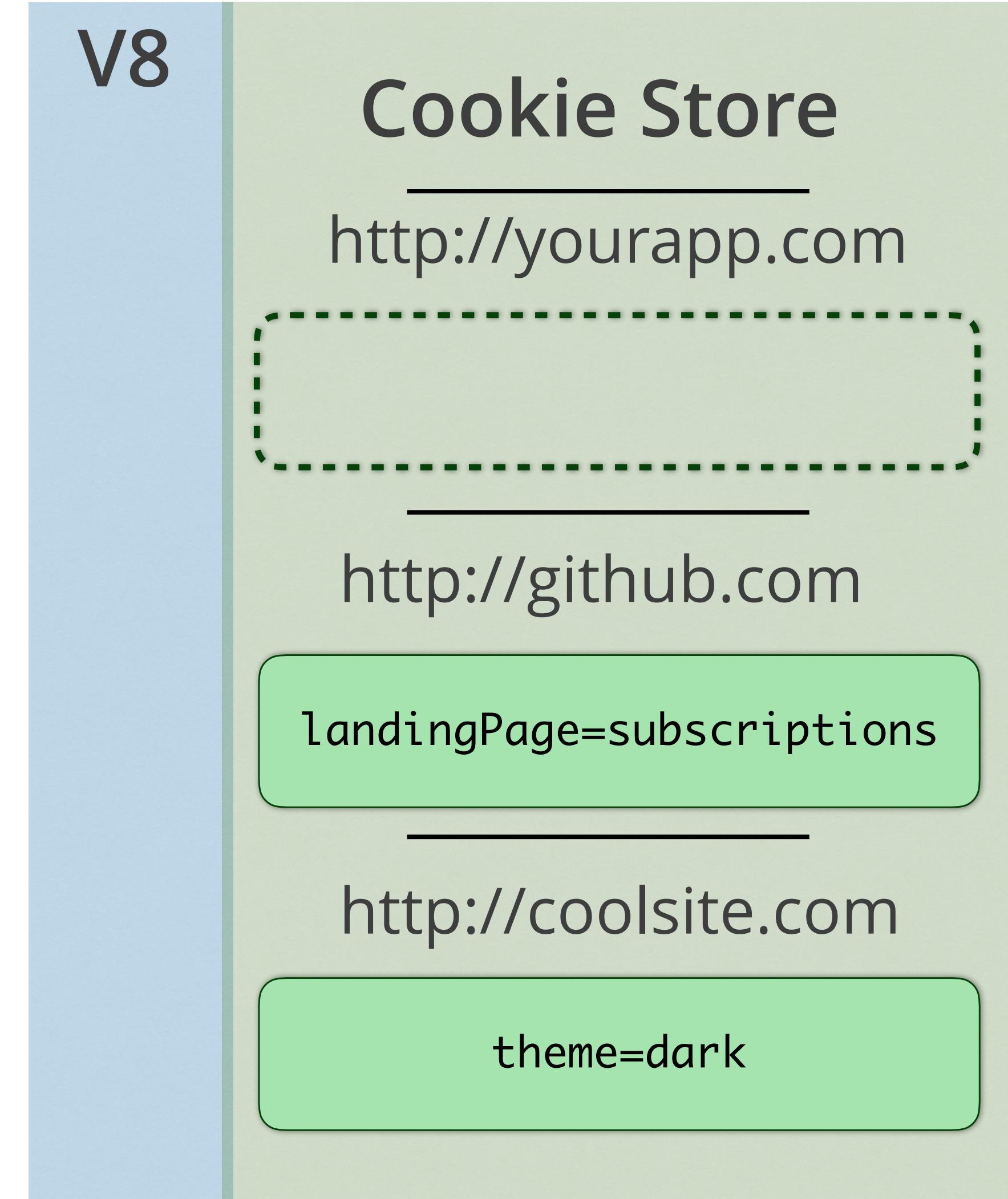


Internet (HTTP)

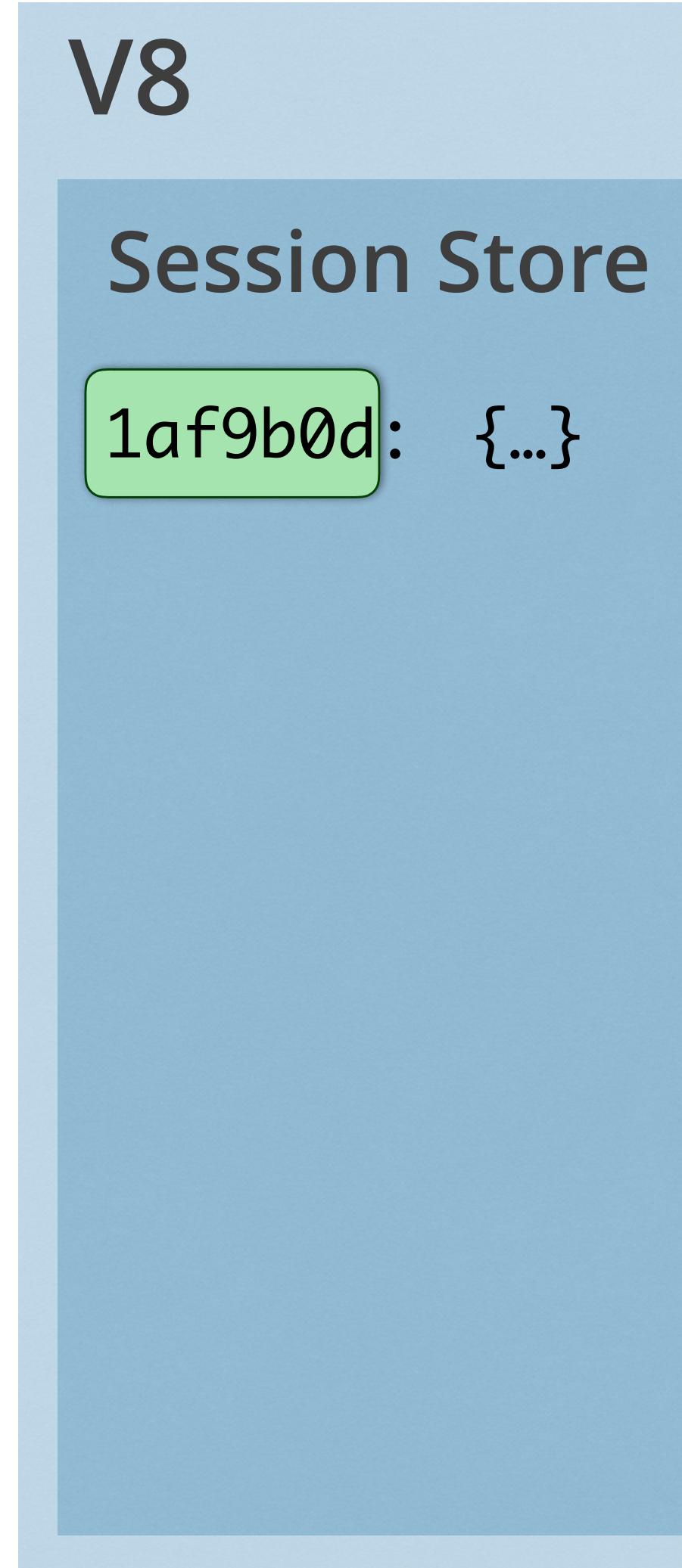


<http://yourapp.com>

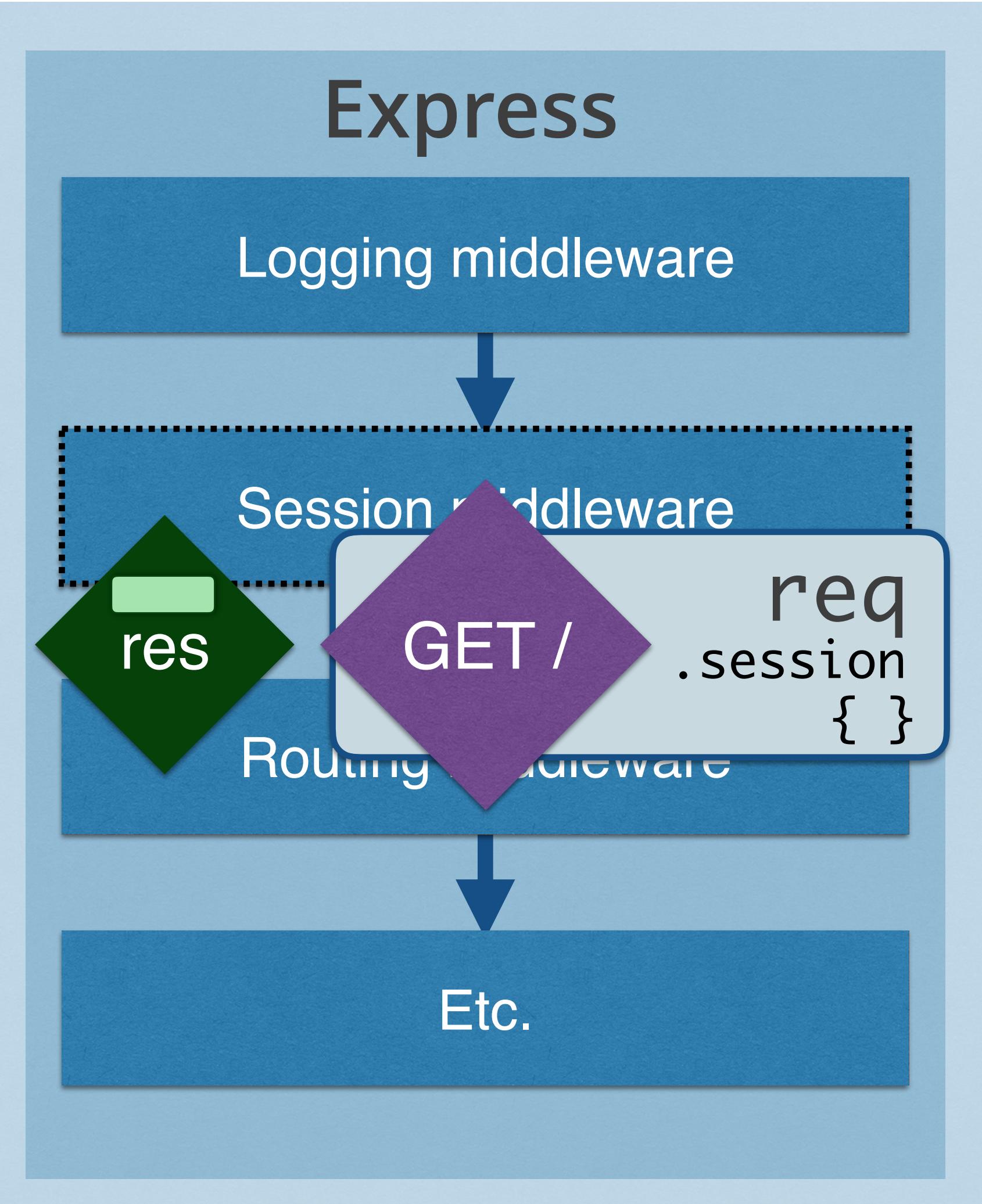
Client (Browser)



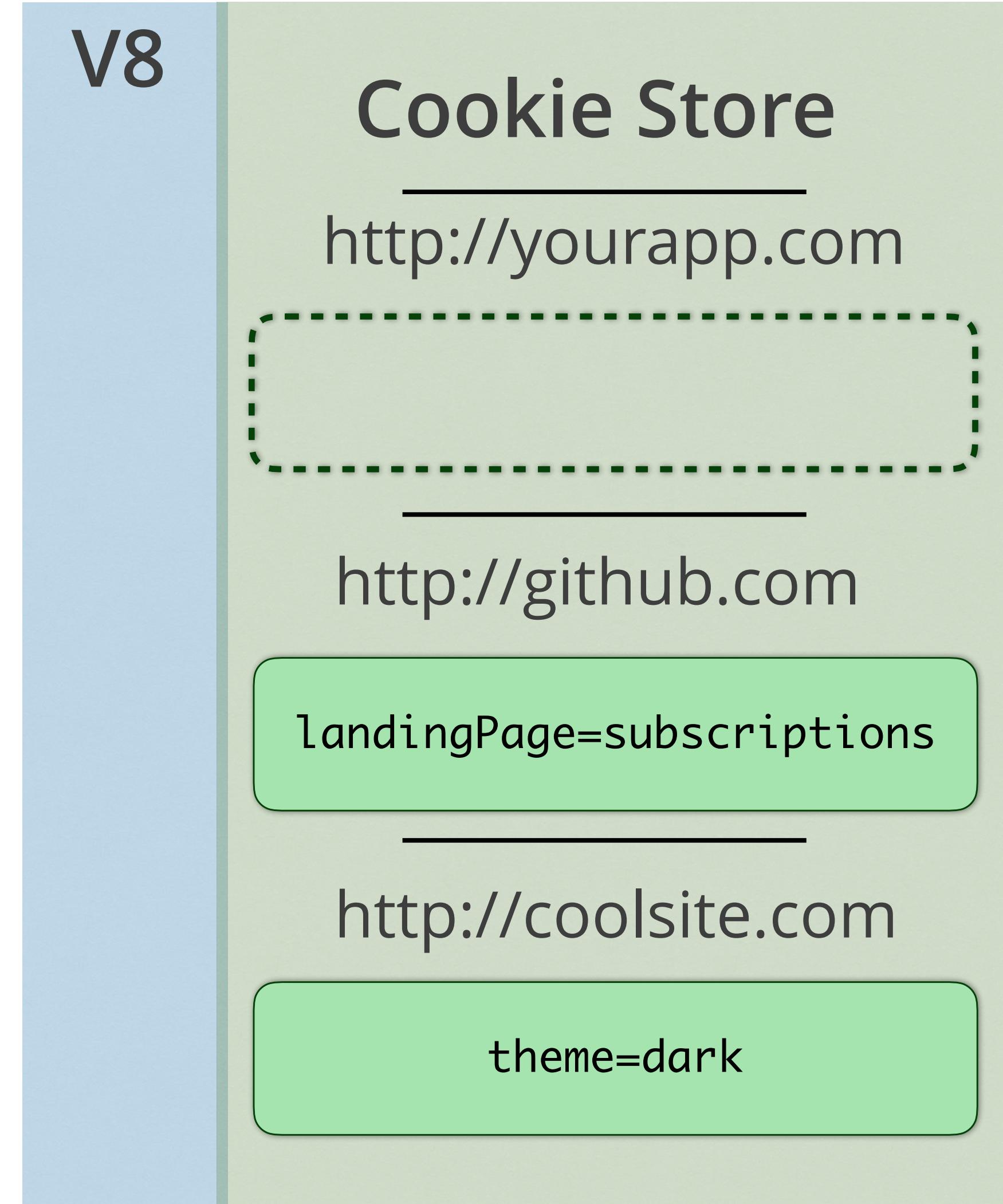
Server (Backend)



Internet (HTTP)

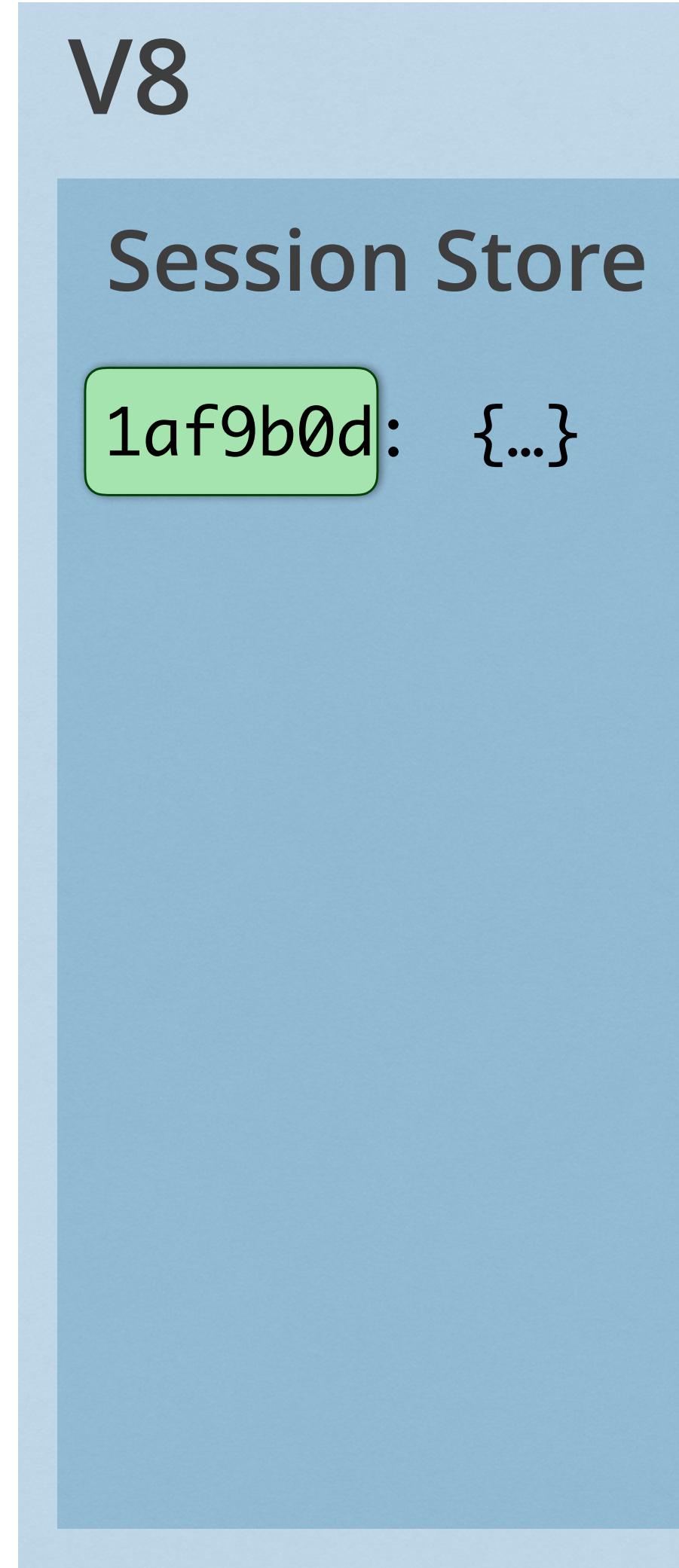


Client (Browser)



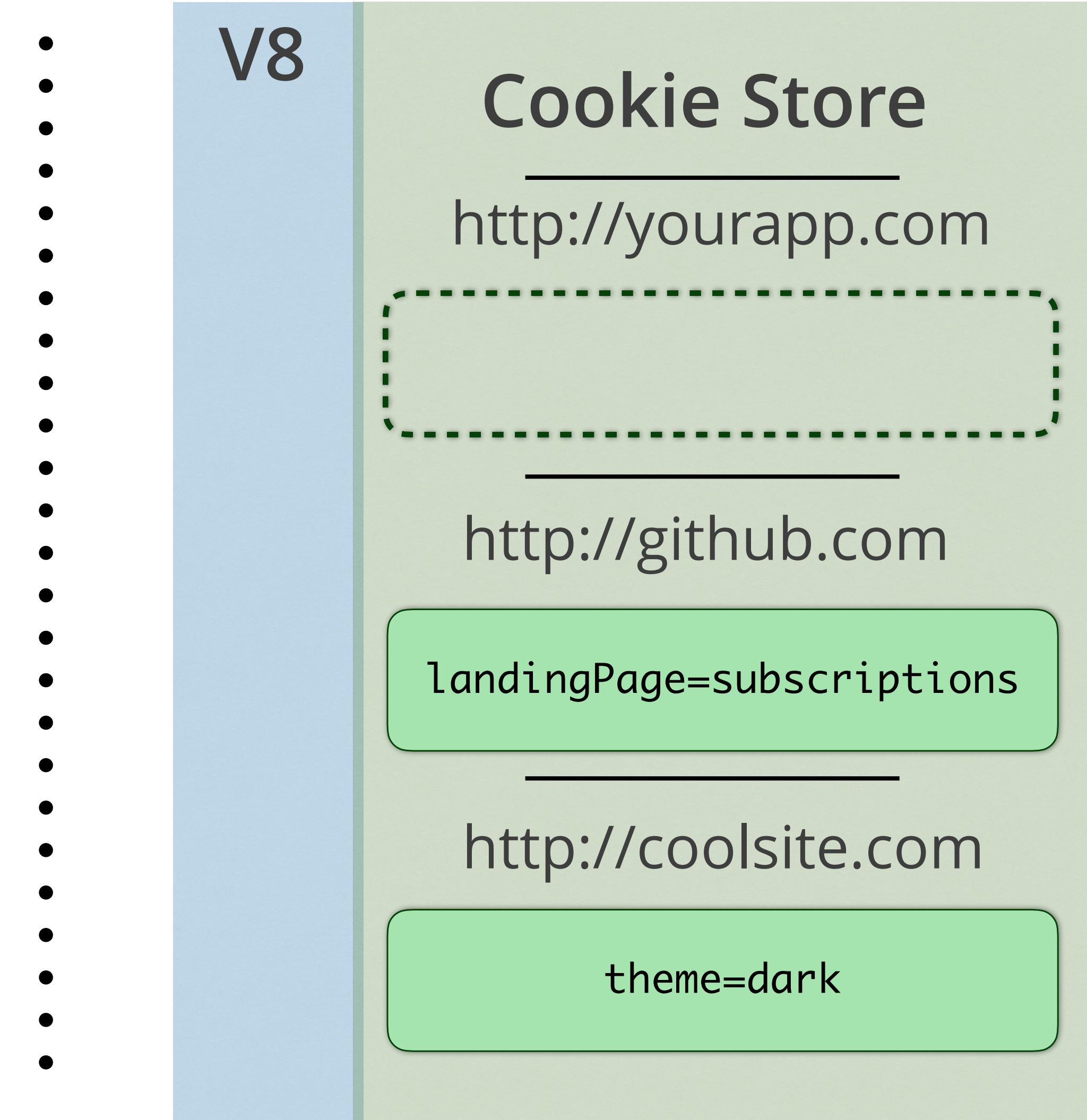
http://yourapp.com

Server (Backend)

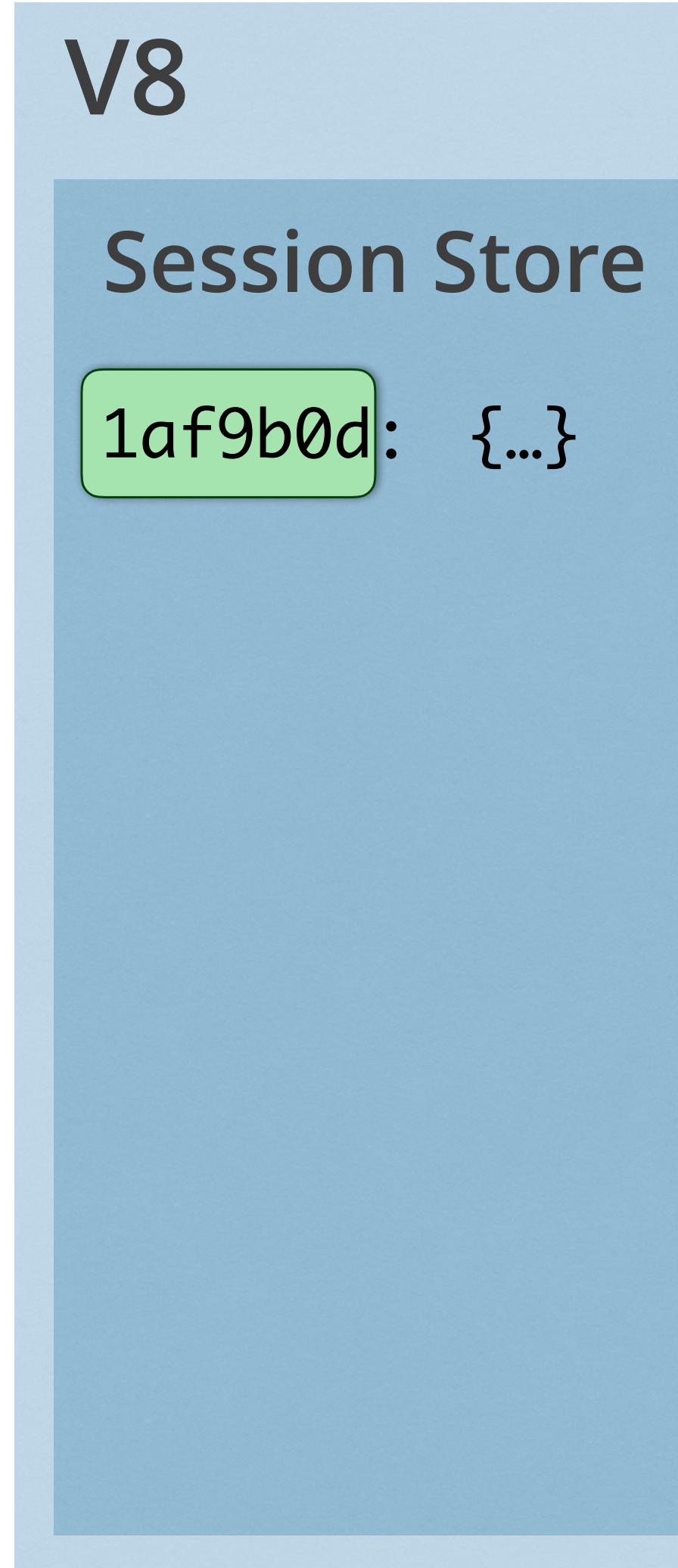


http://yourapp.com

Internet (HTTP)

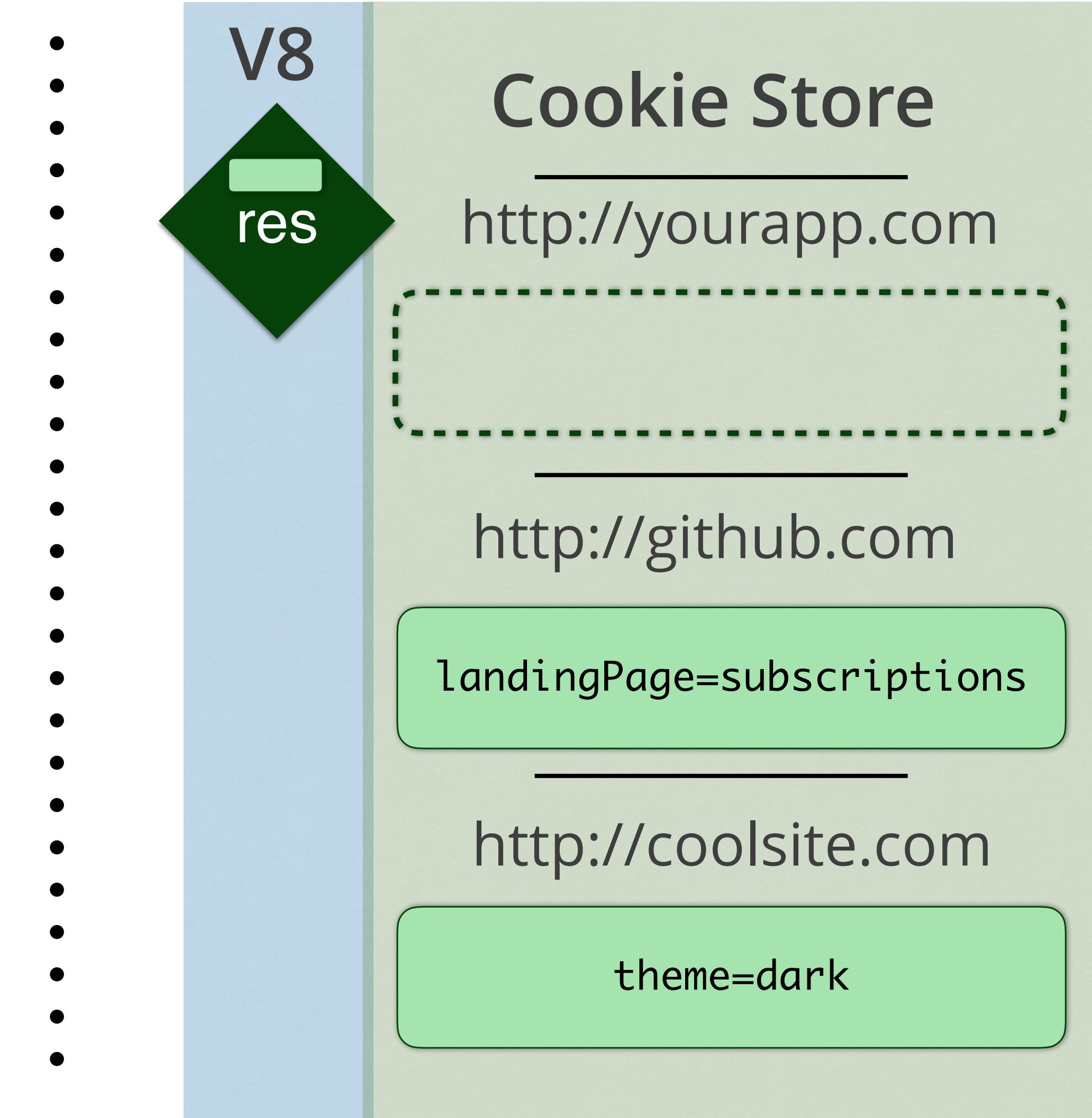


Server (Backend)



<http://yourapp.com>

Internet (HTTP)



HTTP RESPONSE

(headers)

HTTP/1.x 200 OK
Transfer-Encoding: chunked
Date: Mon, 22 Feb 2016 18:30:00 GMT
Content-Type: text/html
Content-Encoding: gzip
set-cookie: myUid=1af9b0d

(body, after
routing is
done)

```
<!DOCTYPE html>
<html>
  <head>
    <title>Your Sweet App</title>
    <script src="/js/main.js"></script>
  </head> ...etc.
```

HTTP RESPONSE

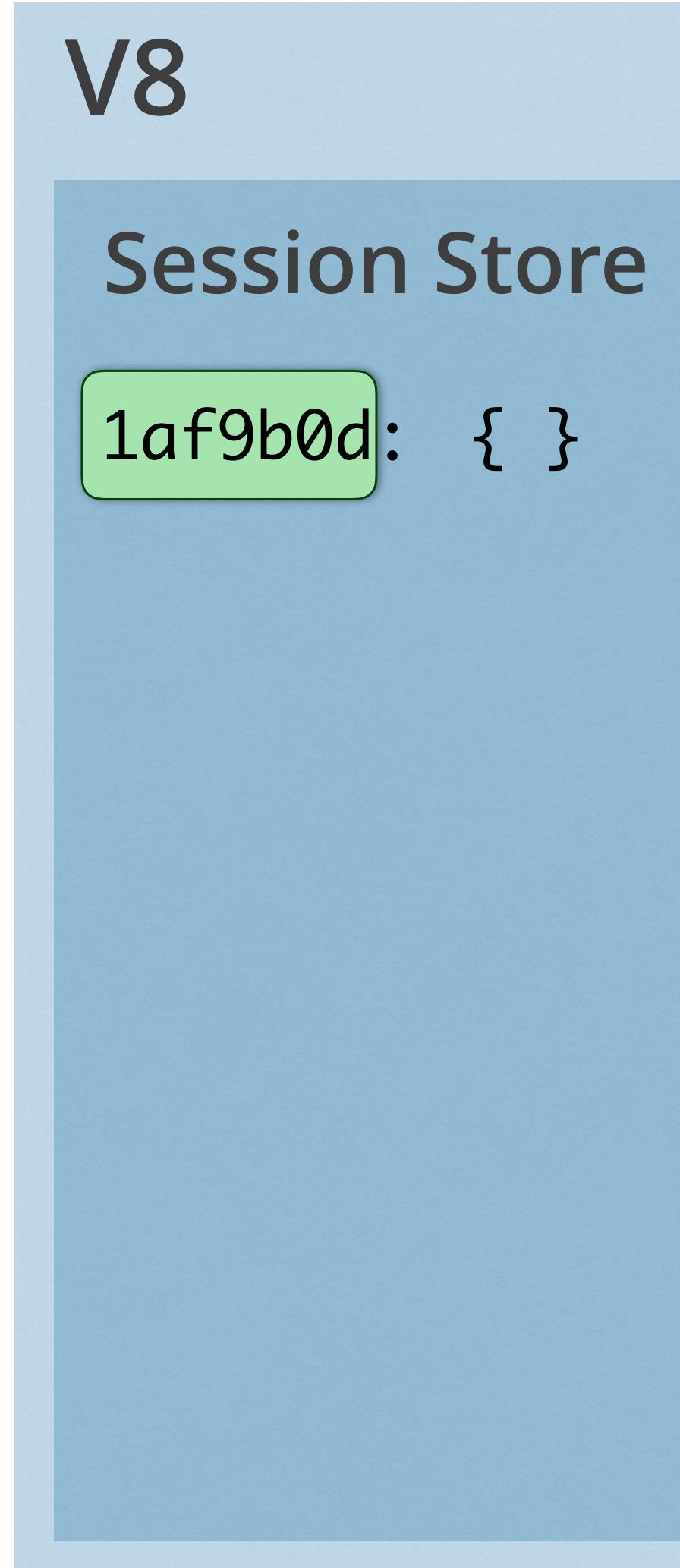
(headers)

HTTP/1.x 200 OK
Transfer-Encoding: chunked
Date: Mon, 22 Feb 2016 18:30:00 GMT
Content-Type: text/html
Content-Encoding: gzip
set-cookie: myUid=1af9b0d

(body, after
routing is
done)

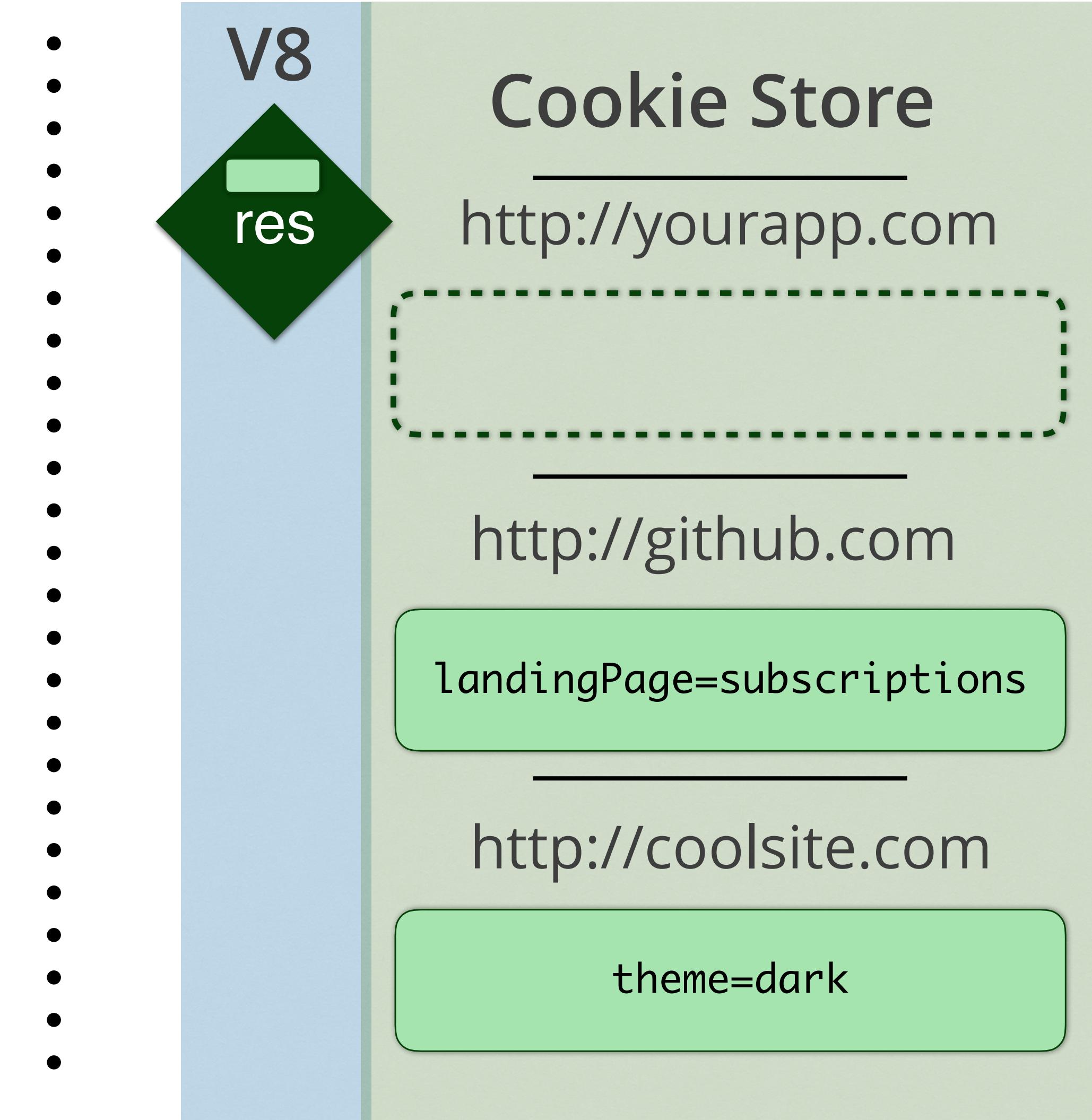
```
<!DOCTYPE html>
<html>
  <head>
    <title>Your Sweet App</title>
    <script src="/js/main.js"></script>
  </head> ...etc.
```

Server (Backend)

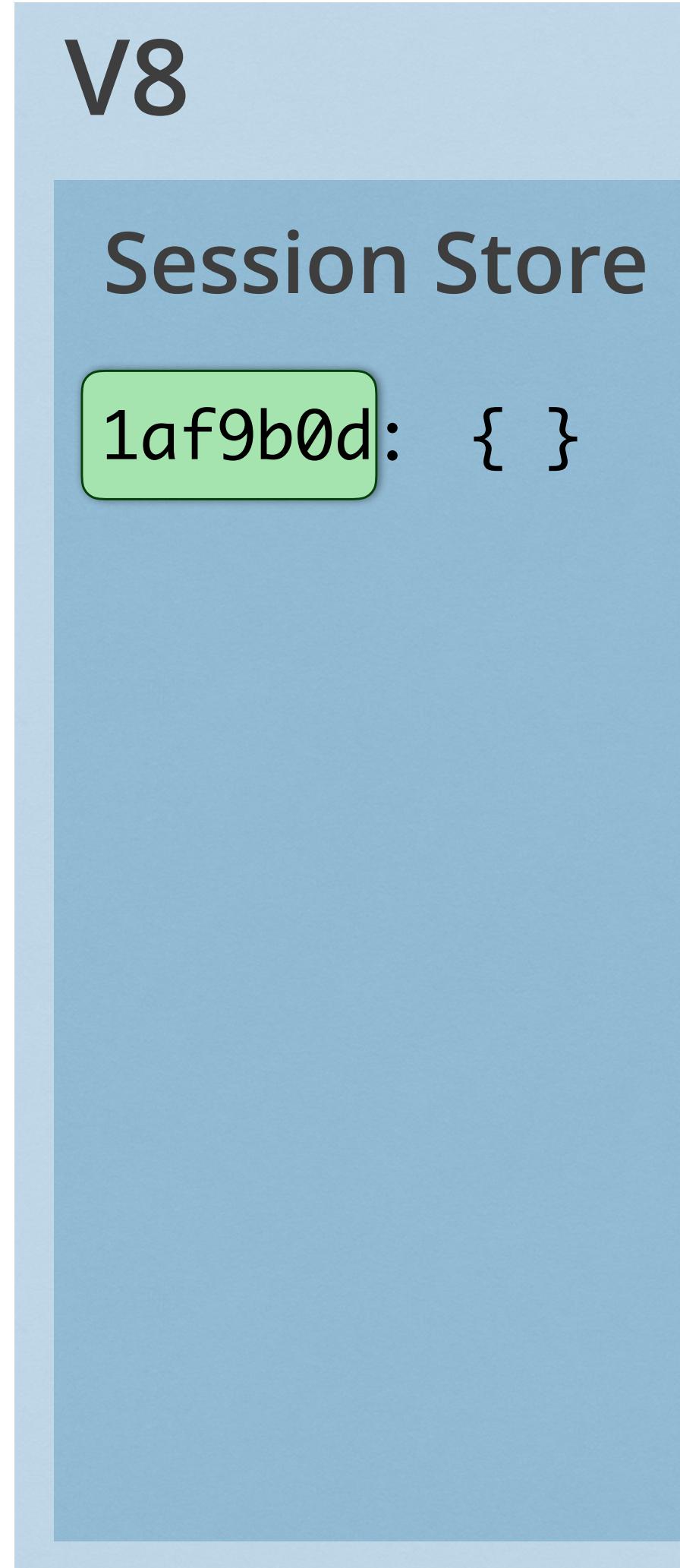


<http://yourapp.com>

Internet (HTTP)

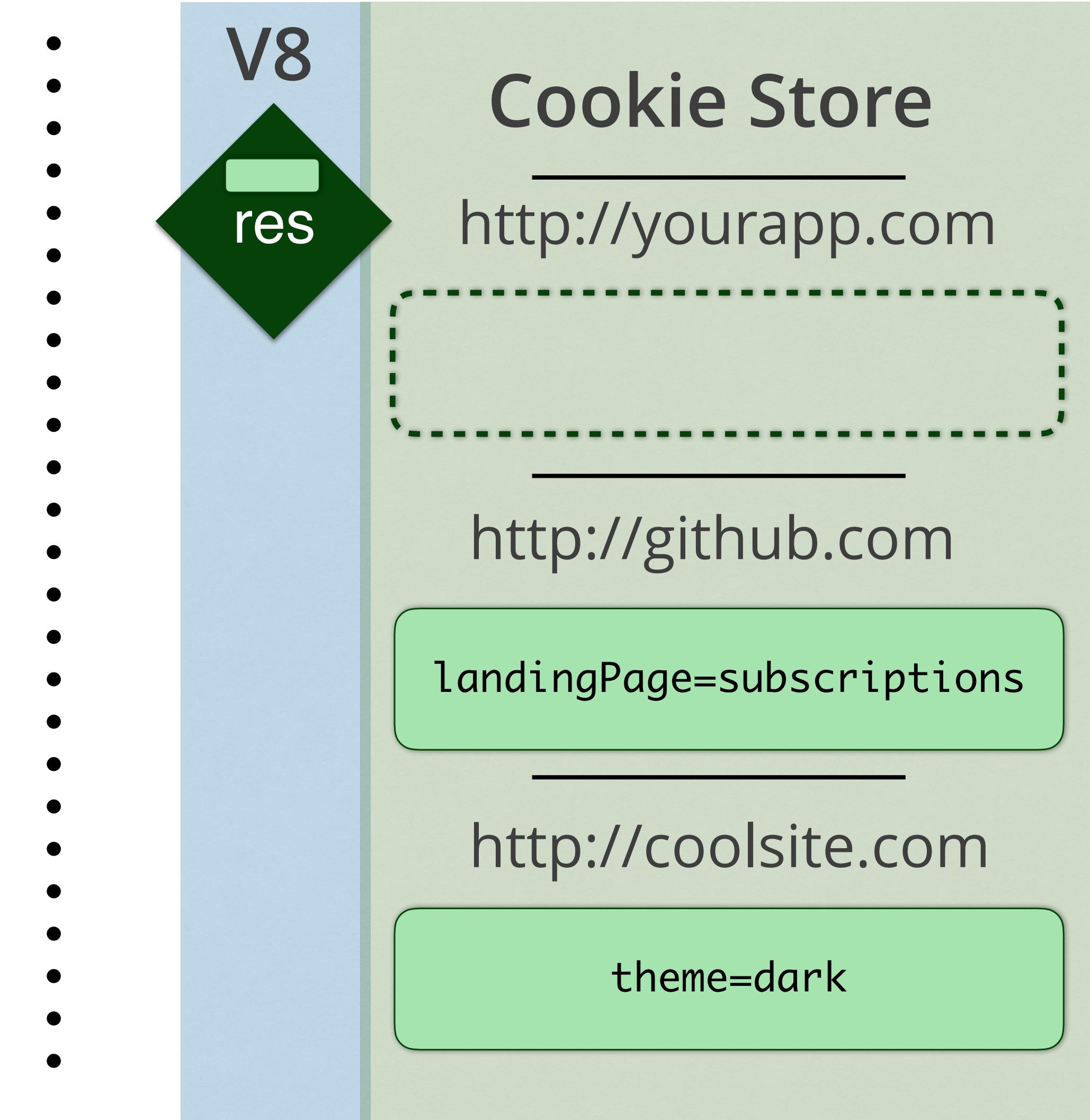


Server (Backend)

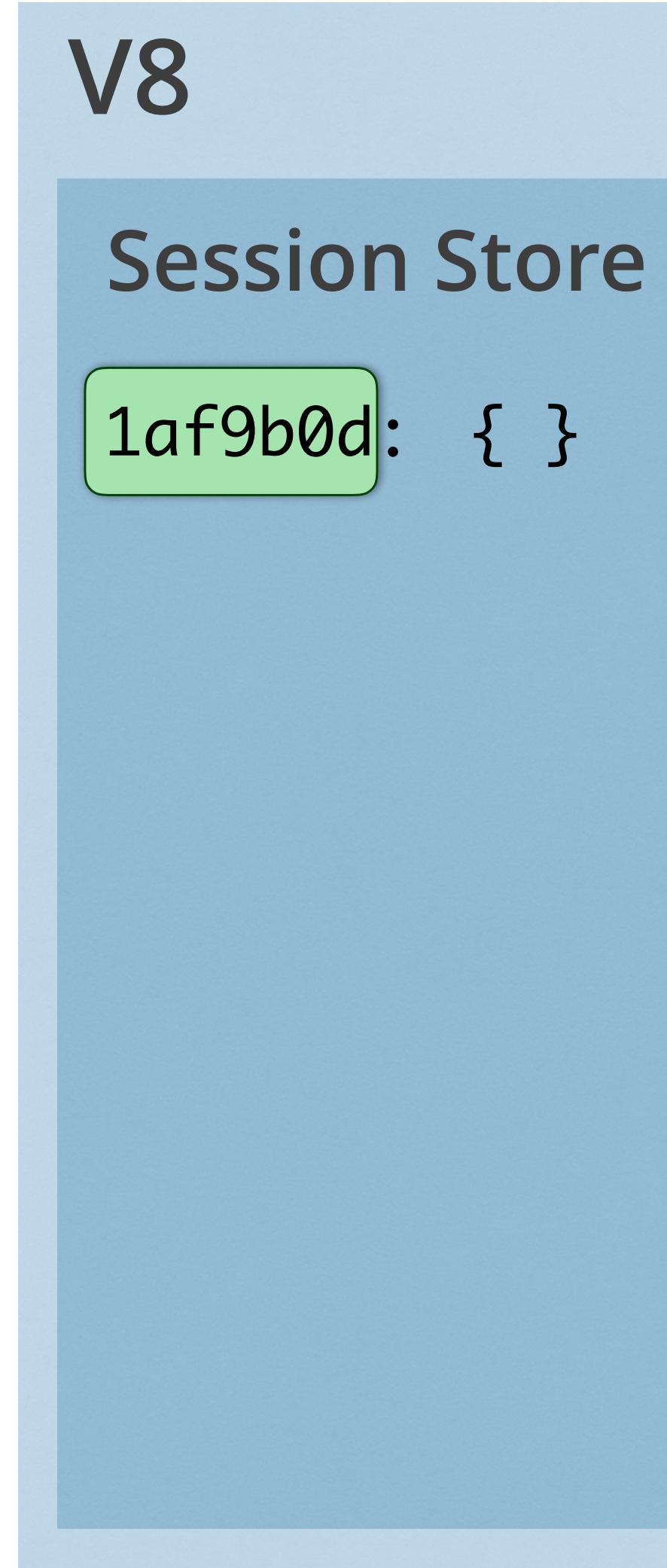


<http://yourapp.com>

Internet (HTTP)



Server (Backend)

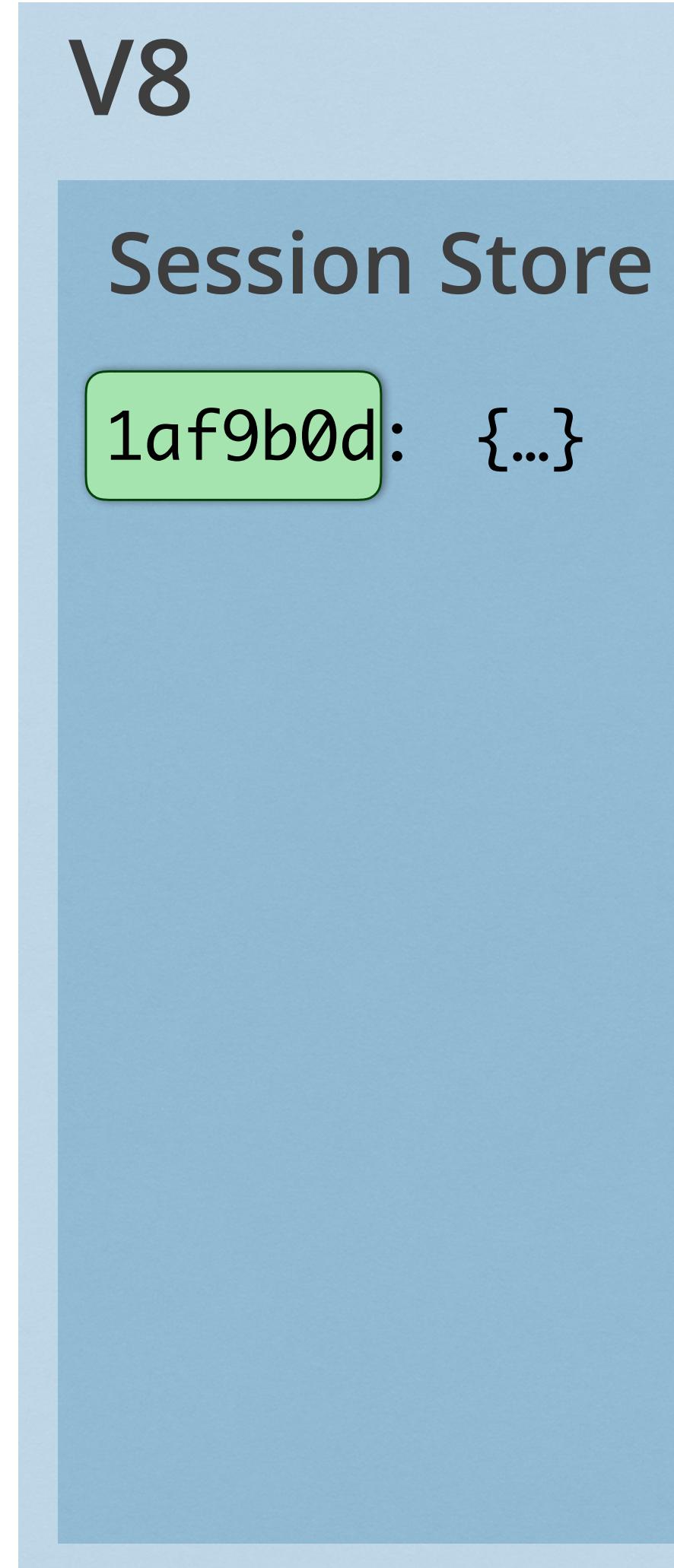


<http://yourapp.com>

Internet (HTTP)



Server (Backend)

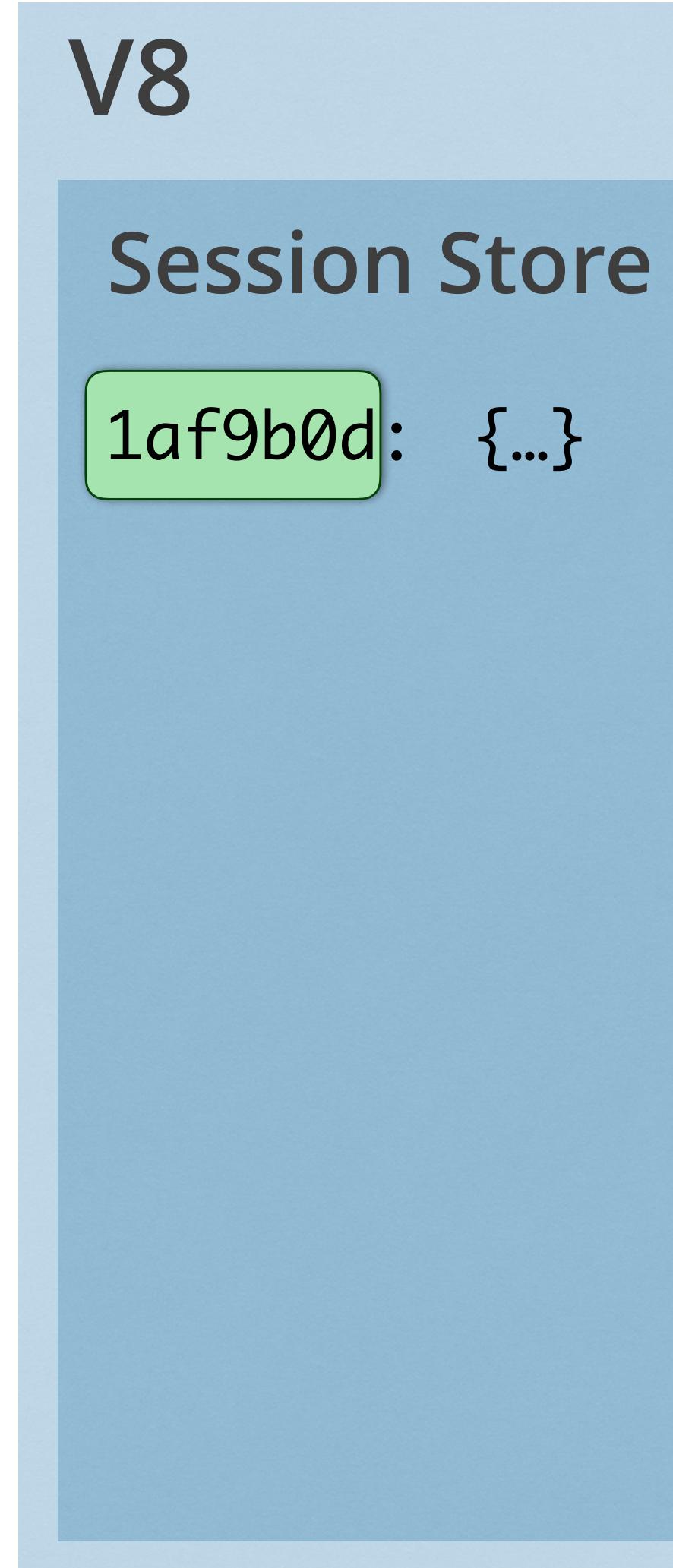


<http://yourapp.com>

Internet (HTTP)

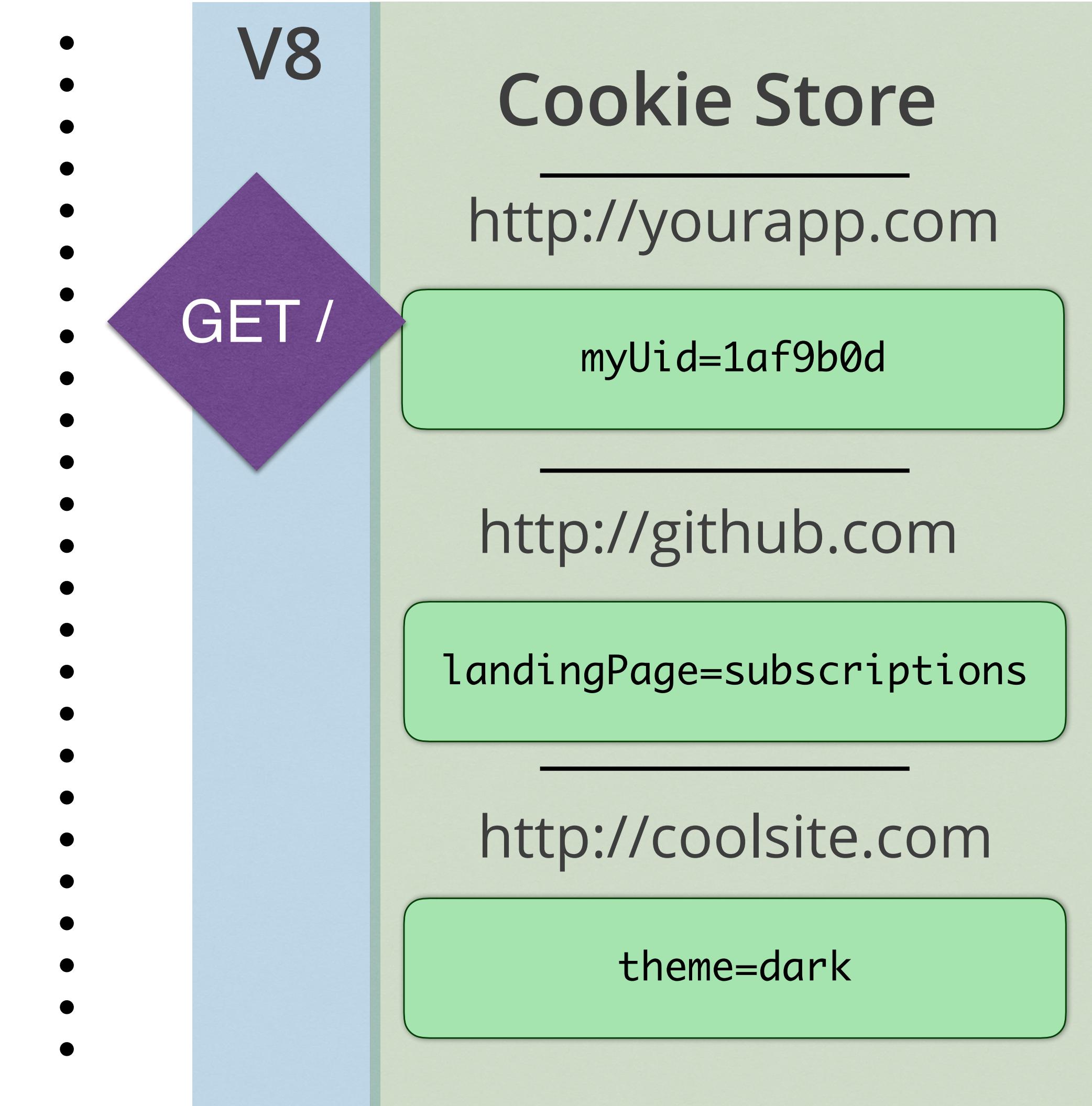


Server (Backend)



<http://yourapp.com>

Internet (HTTP)



HTTP REQUEST

(headers)

```
GET / HTTP/1.1
Host: yourapp.com
Connection: keep-alive
Accept: text/html
User-Agent: Chrome/
48.0.2564.116
```

(body)

HTTP REQUEST

(headers)

(body)

GET / HTTP/1.1
Host: yourapp.com
Connection: keep-alive
Accept: text/html
User-Agent: Chrome/
48.0.2564.116
Cookie: myUid=1af9b0d

HTTP REQUEST

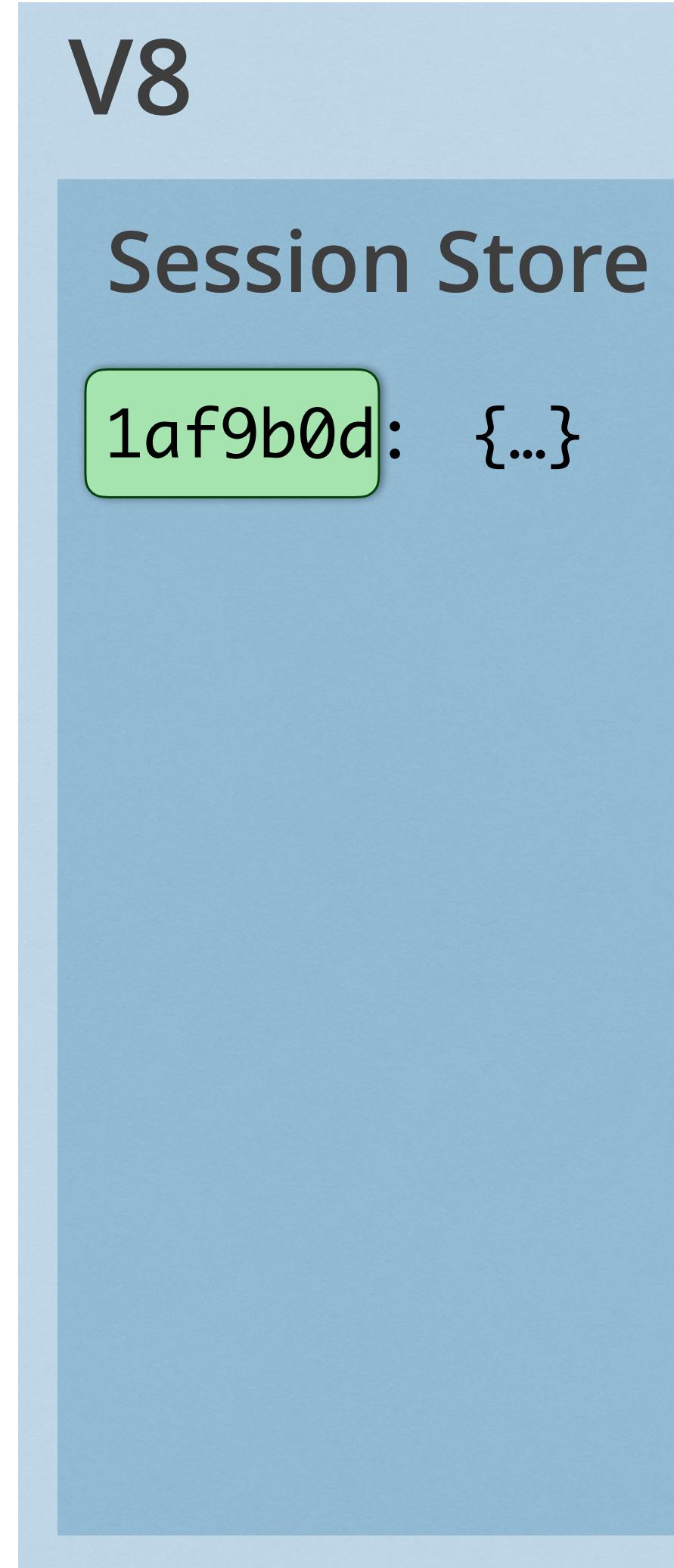
(headers)

(body)

GET / HTTP/1.1
Host: yourapp.com
Connection: keep-alive
Accept: text/html
User-Agent: Chrome/
48.0.2564.116
Cookie: myUid=1af9b0d

...attached cookie!

Server (Backend)



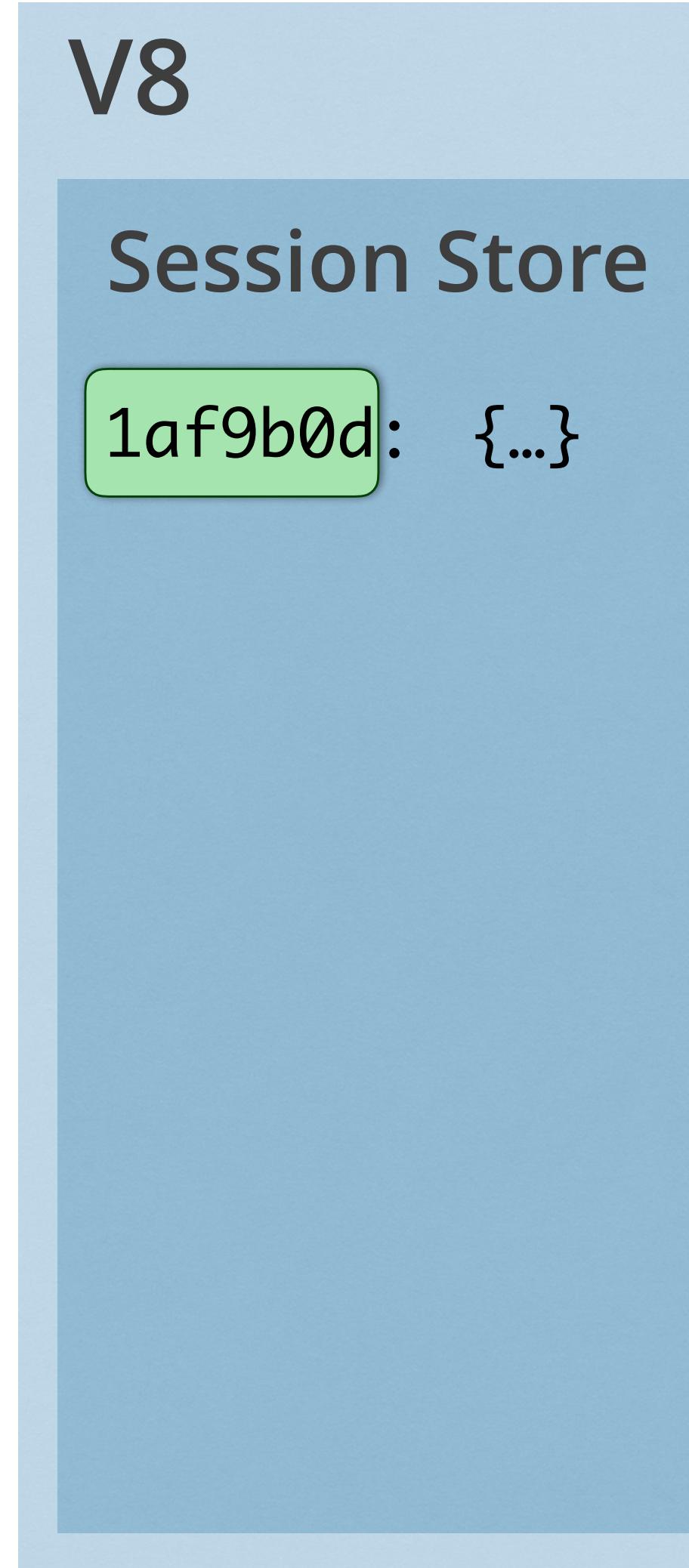
<http://yourapp.com>

Internet (HTTP)



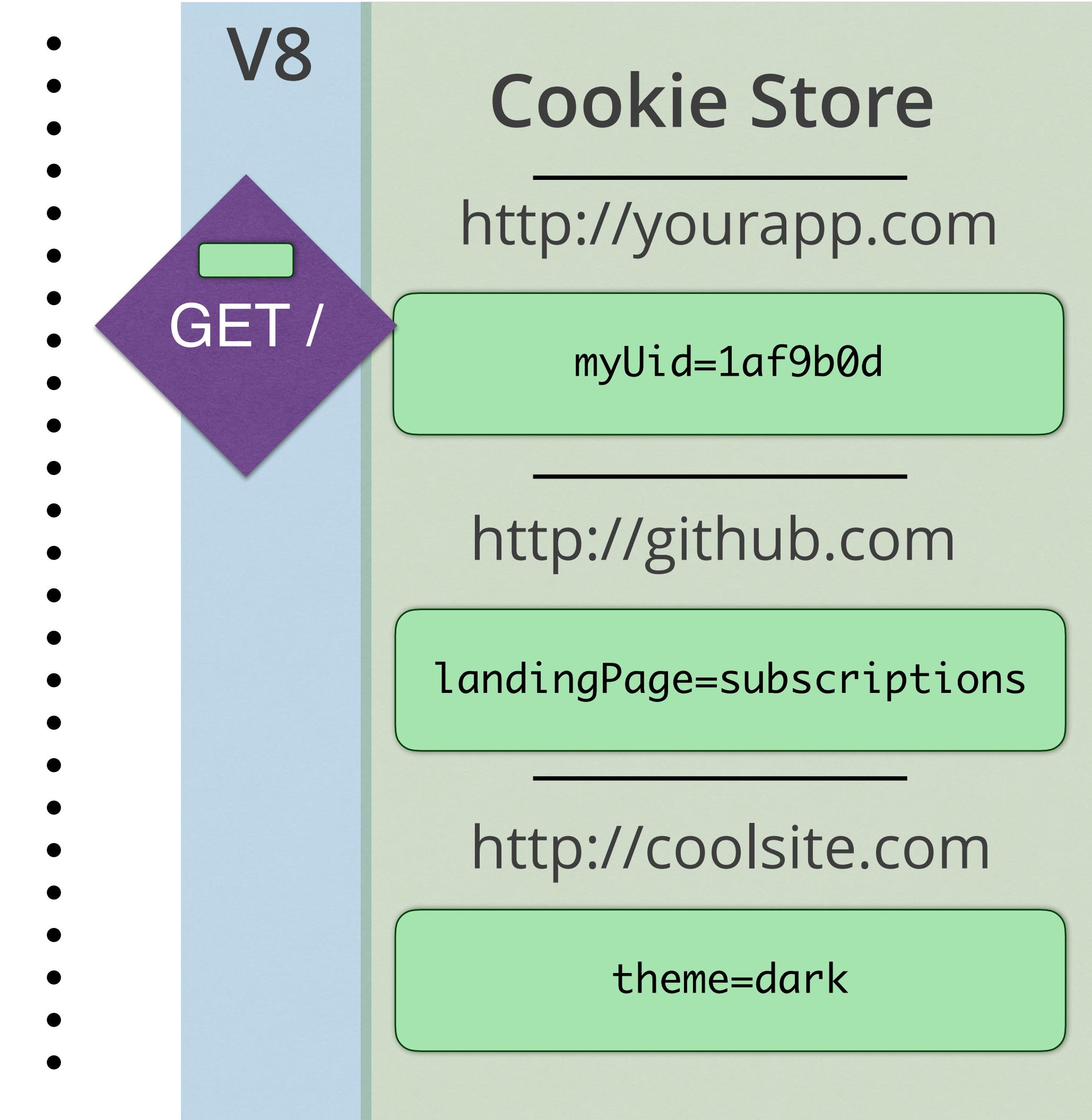
AUTH

Server (Backend)



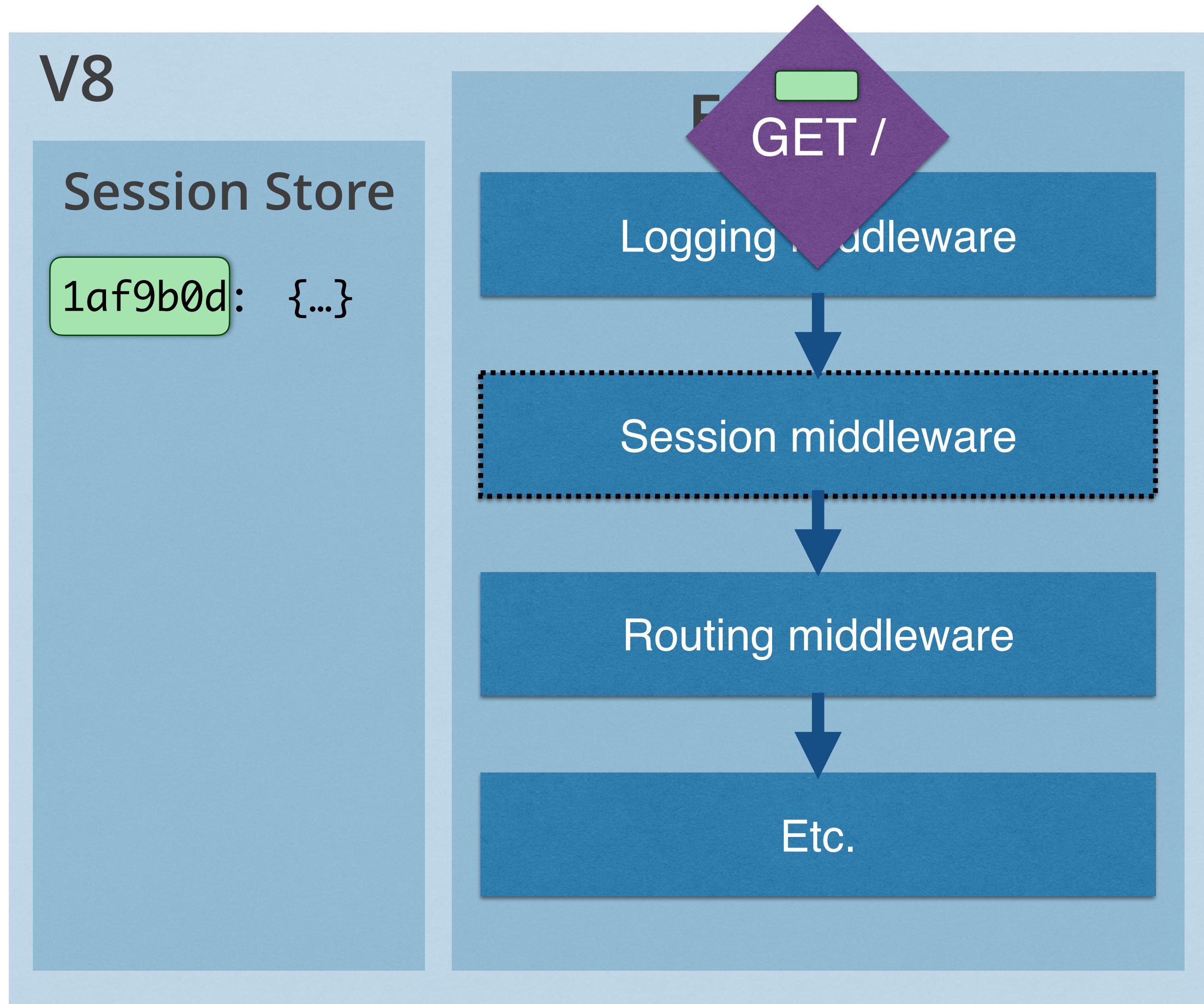
<http://yourapp.com>

Internet (HTTP)



Server (Backend)

V8
Session Store
1af9b0d: {...}



<http://yourapp.com>

Internet (HTTP)

⋮
V8

Client (Browser)

Cookie Store
<http://yourapp.com>

myUid=1af9b0d

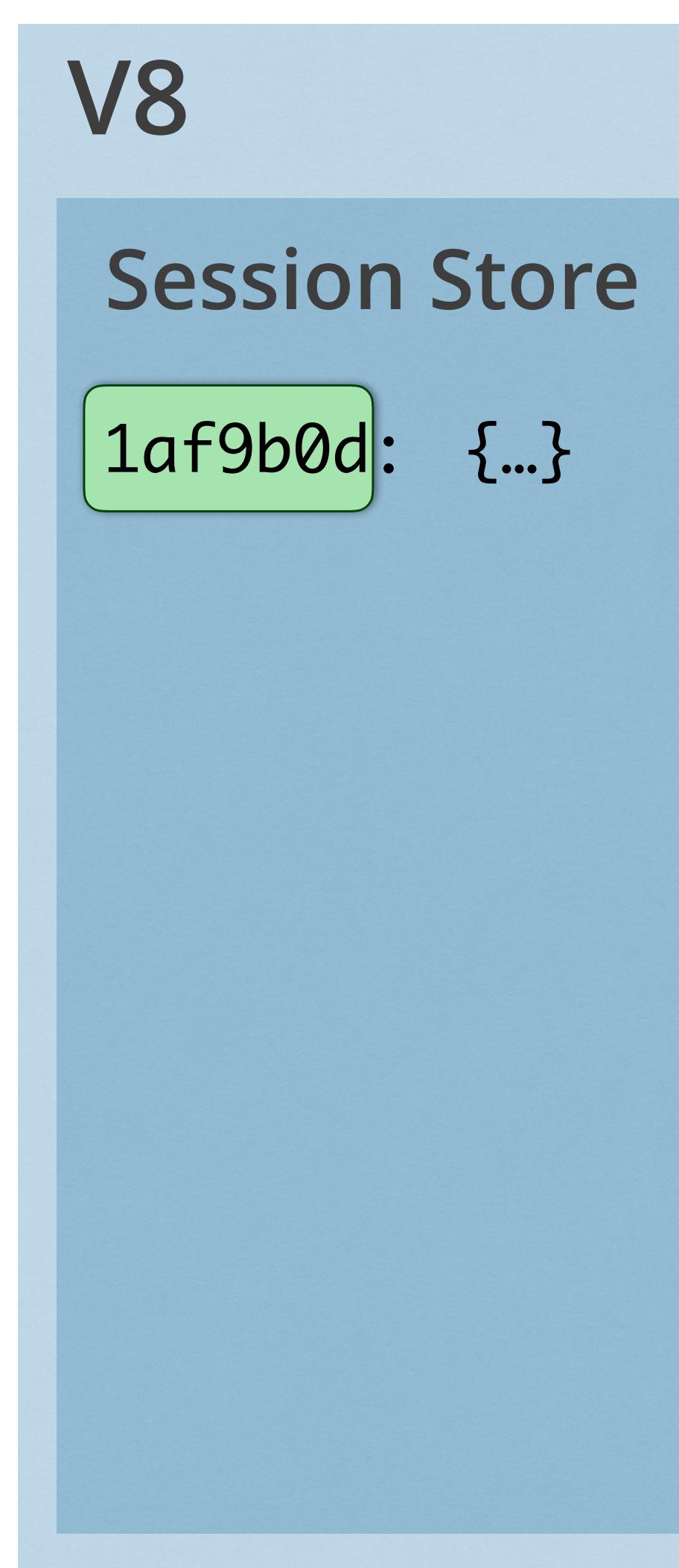
<http://github.com>

landingPage=subscriptions

<http://coolsite.com>

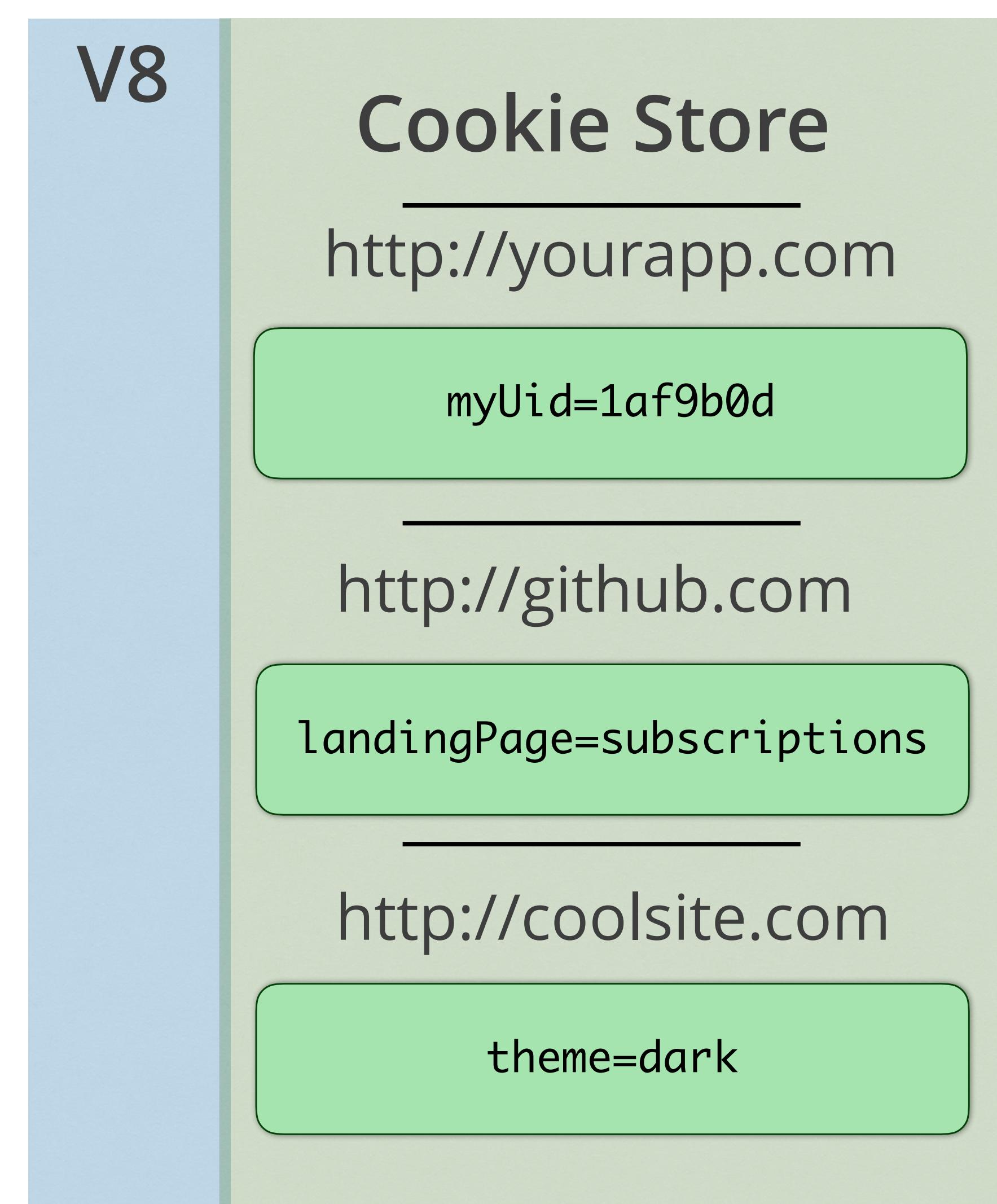
theme=dark

Server (Backend)

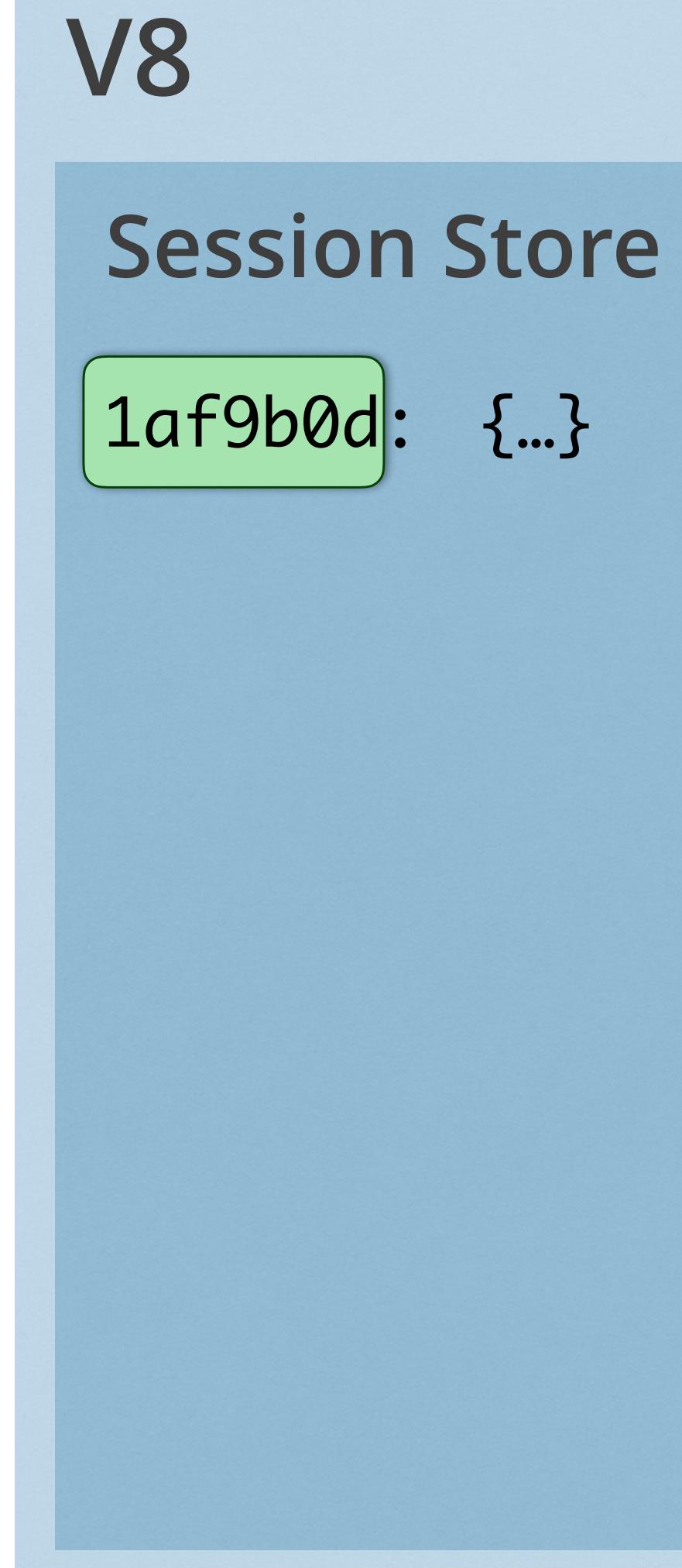


<http://yourapp.com>

Internet (HTTP)



Server (Backend)



<http://yourapp.com>

Internet (HTTP)



HTTP REQUEST

(headers)

(body)

GET / HTTP/1.1
Host: yourapp.com
Connection: keep-alive
Accept: text/html
User-Agent: Chrome/
48.0.2564.116
Cookie: myUid=1af9b0d

HTTP REQUEST

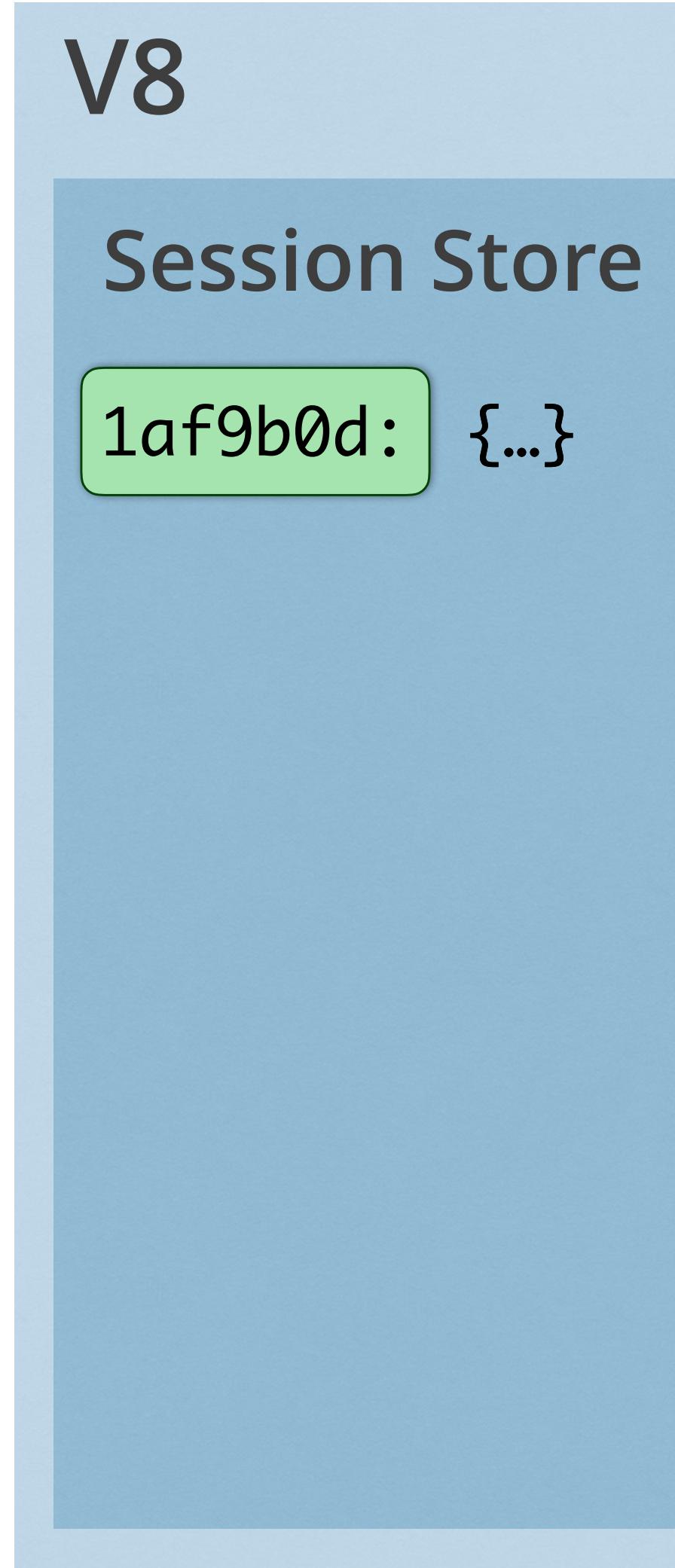
(headers)

(body)

GET / HTTP/1.1
Host: yourapp.com
Connection: keep-alive
Accept: text/html
User-Agent: Chrome/
48.0.2564.116
Cookie: myUid=1af9b0d

attached cookie, so
session middleware:
let's look up the session!

Server (Backend)

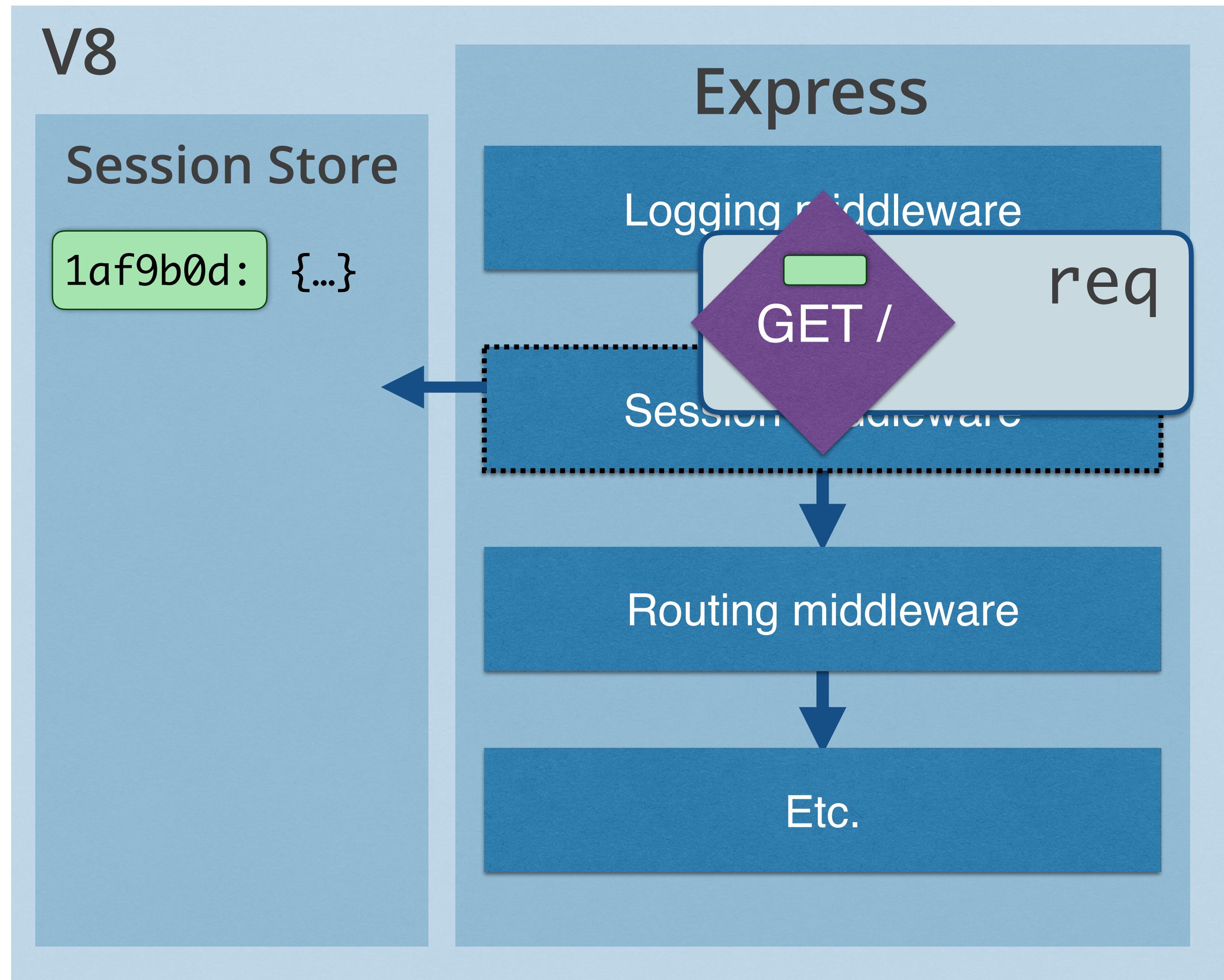


<http://yourapp.com>

Internet (HTTP)

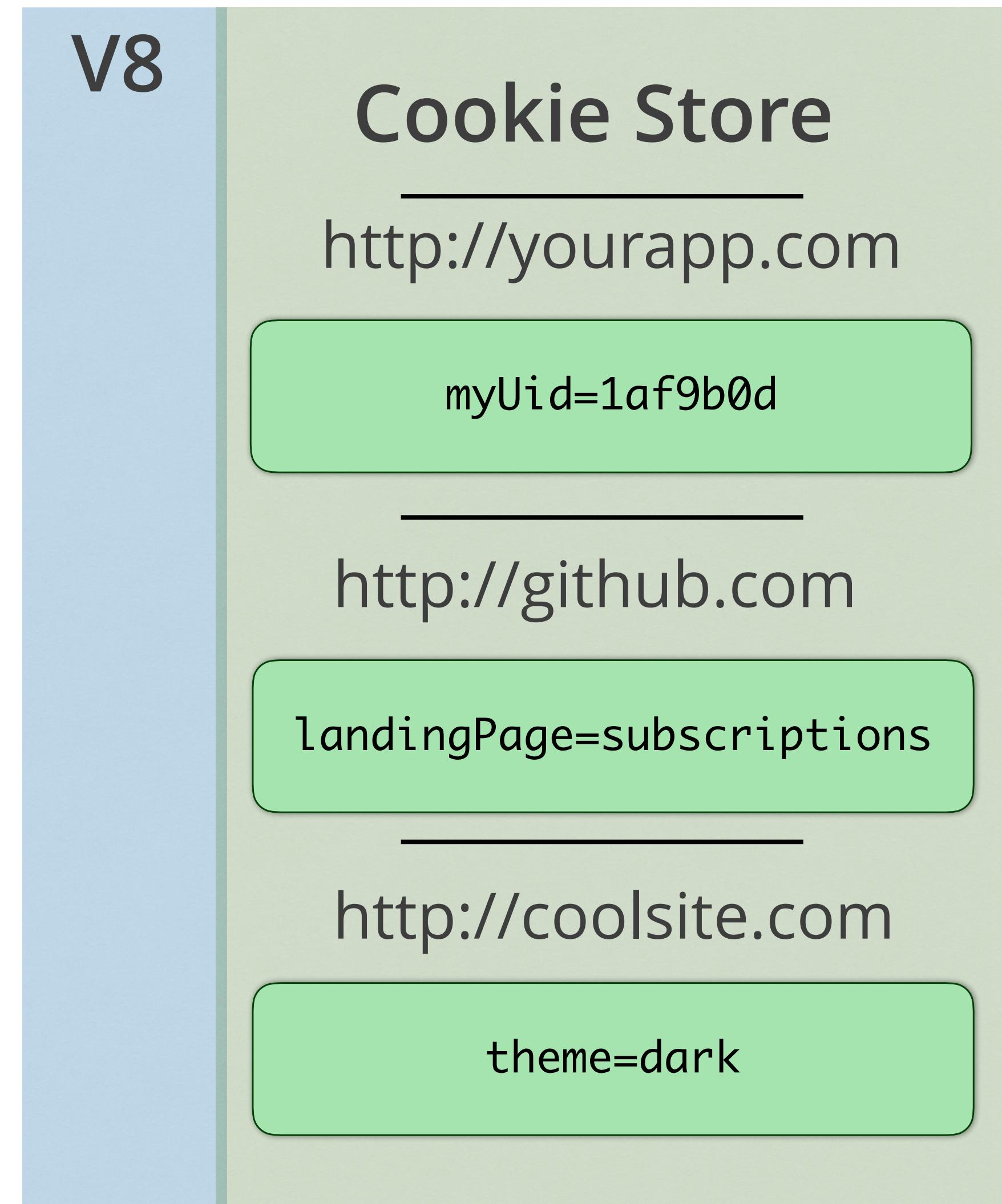


Server (Backend)



<http://yourapp.com>

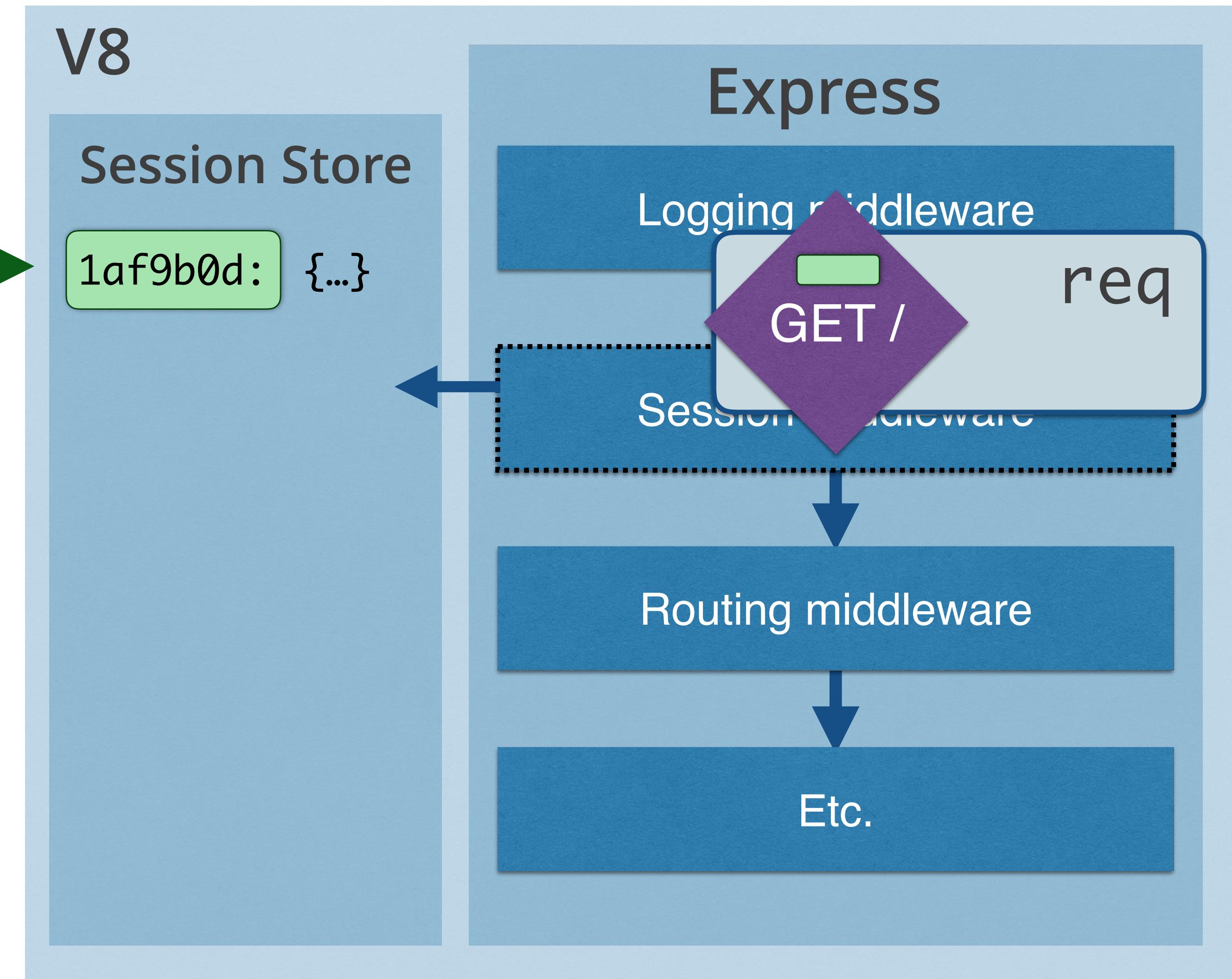
Internet (HTTP)



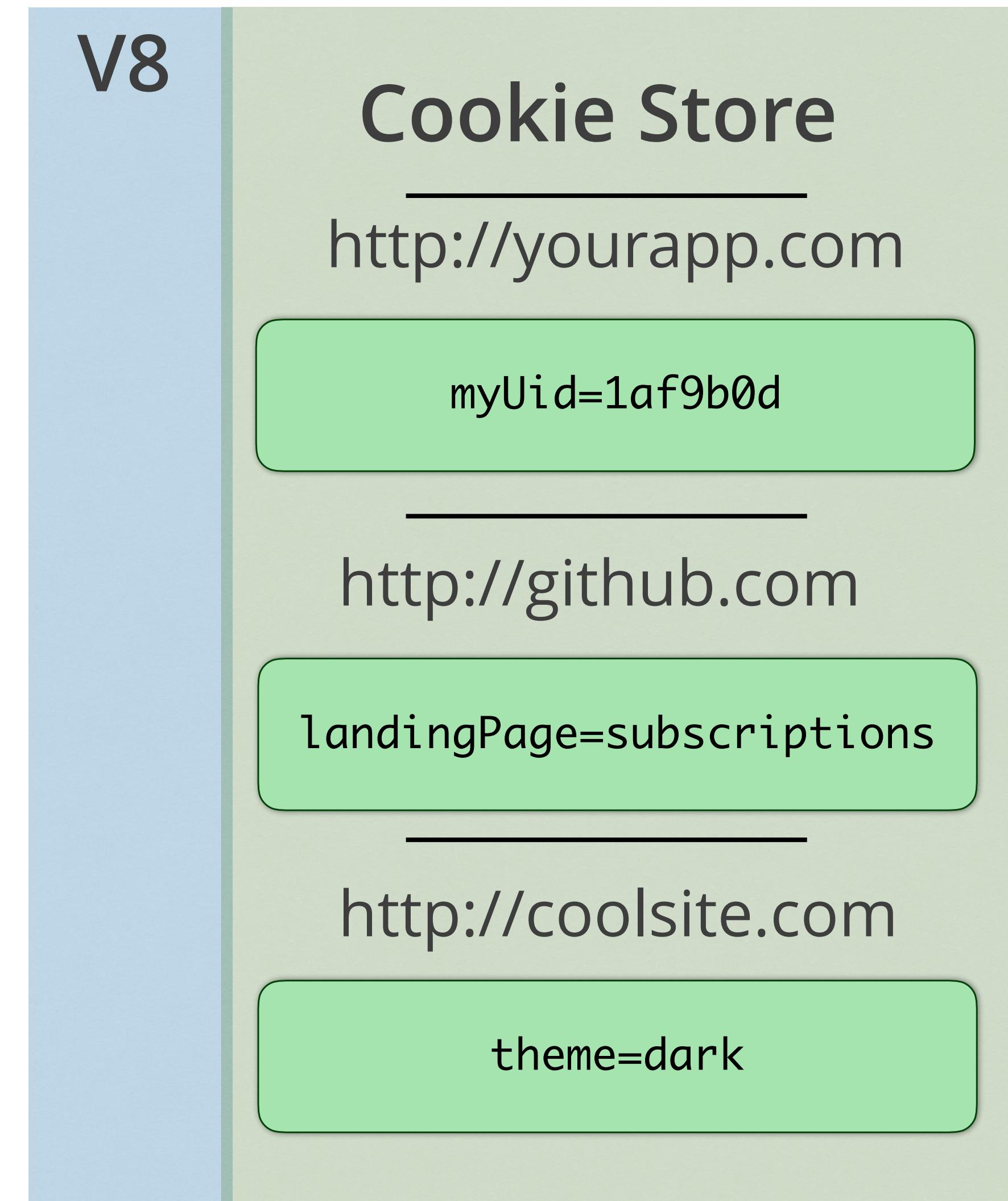
Server (Backend)

Internet (HTTP)

Client (Browser)



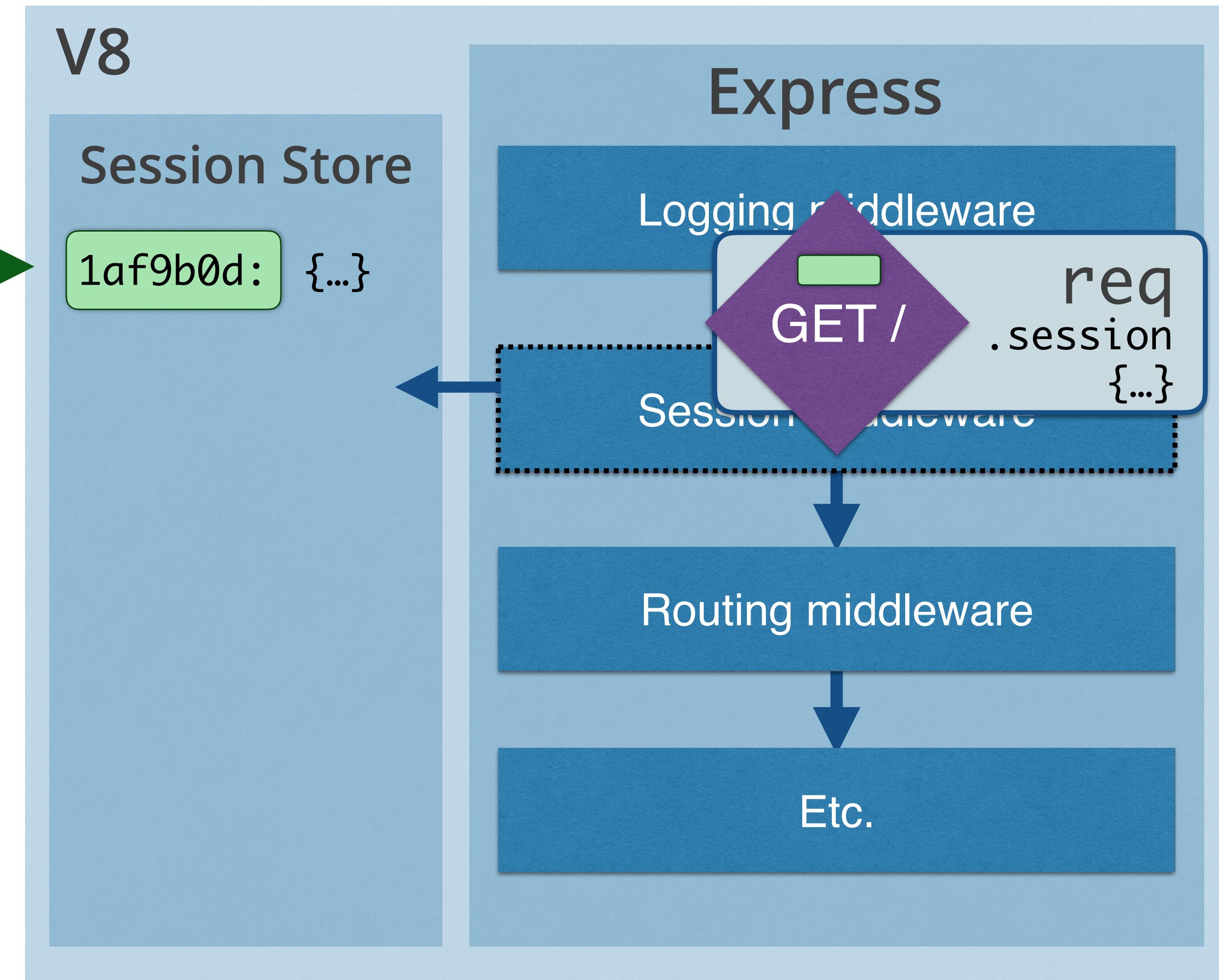
<http://yourapp.com>



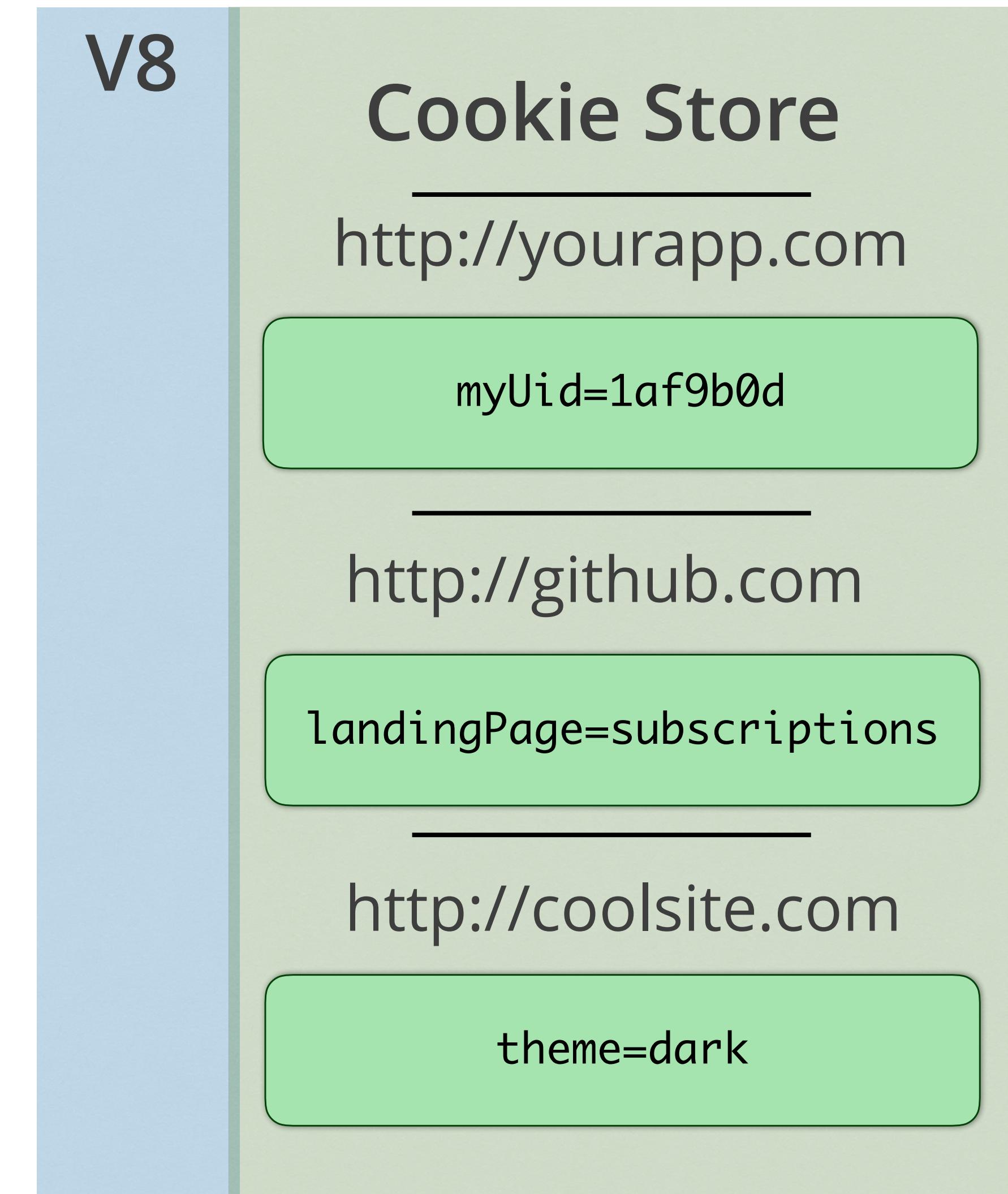
Server (Backend)

Internet (HTTP)

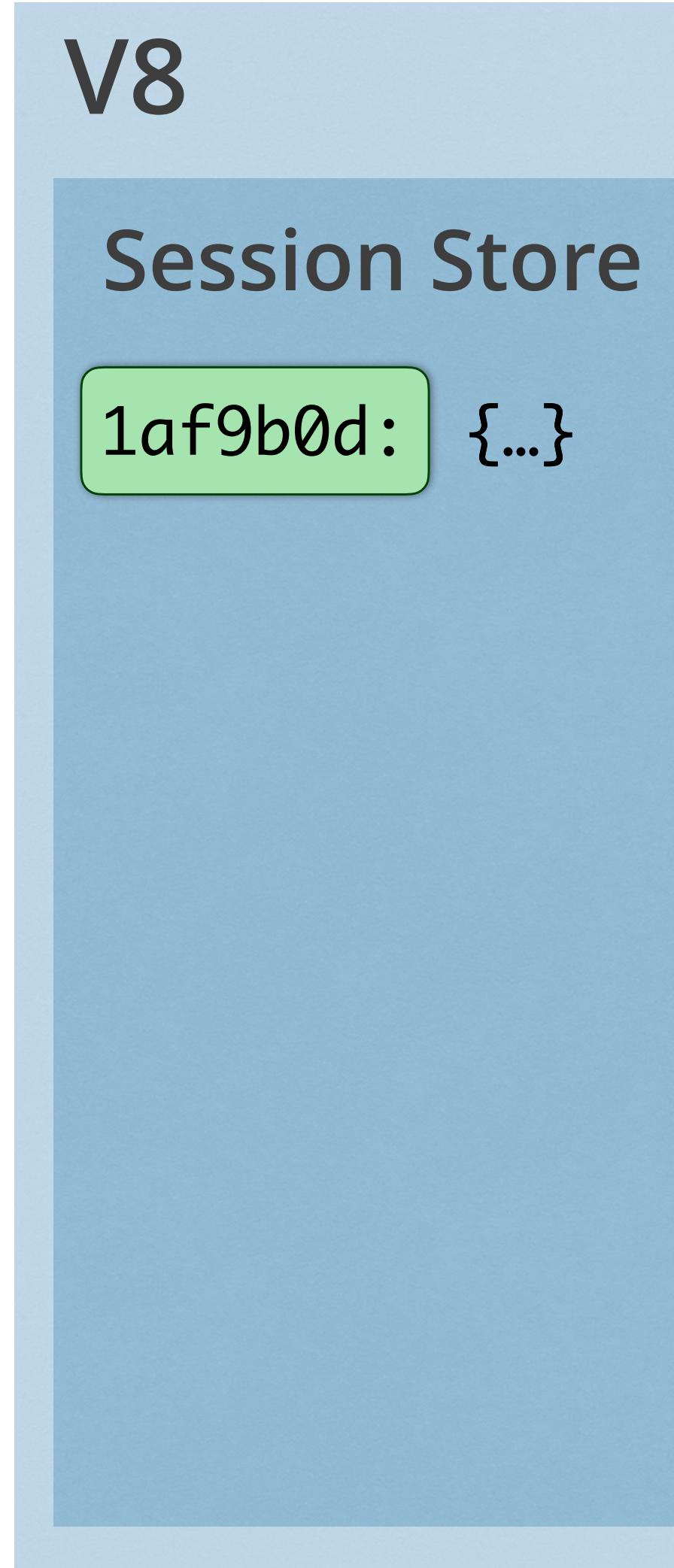
Client (Browser)



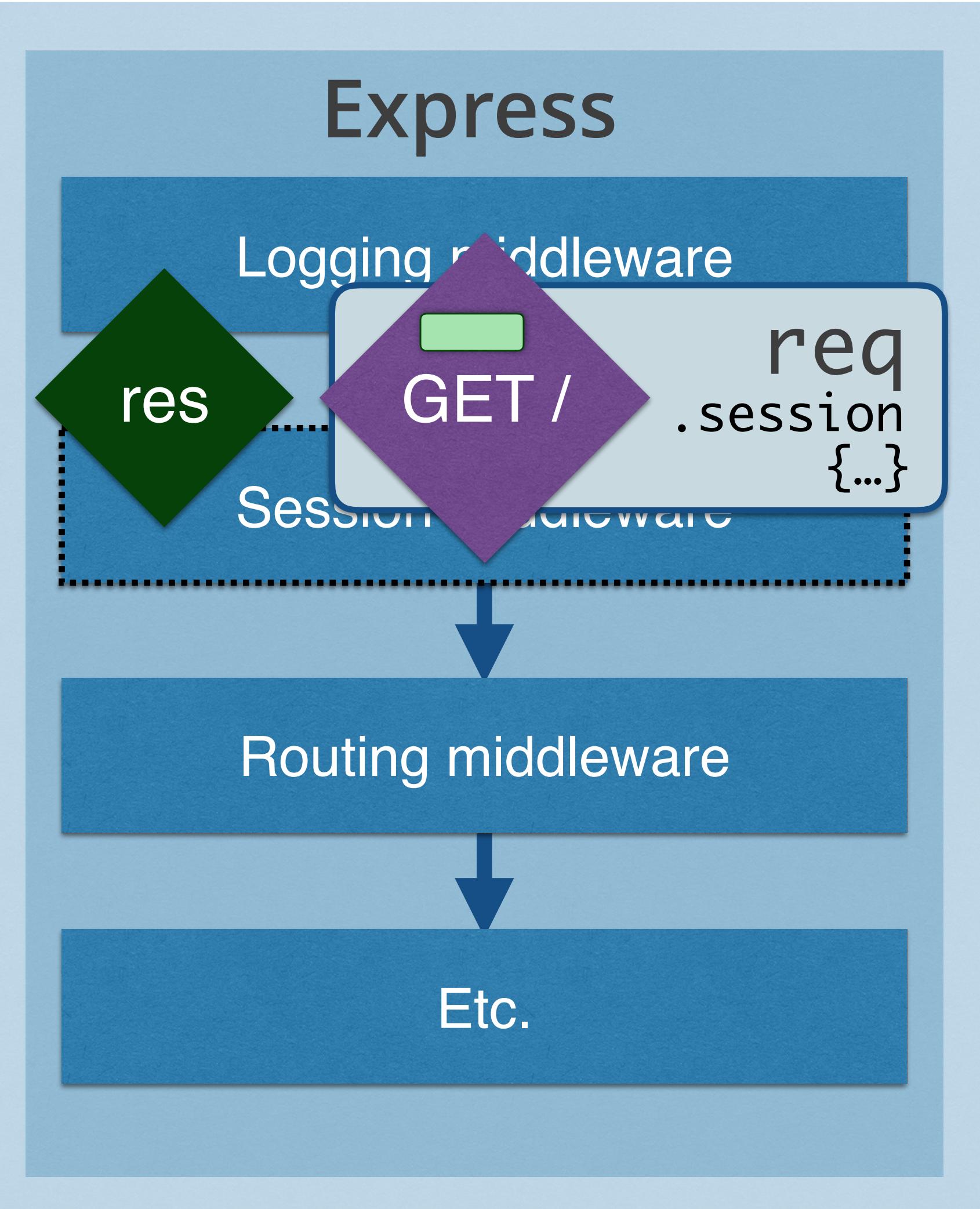
<http://yourapp.com>



Server (Backend)

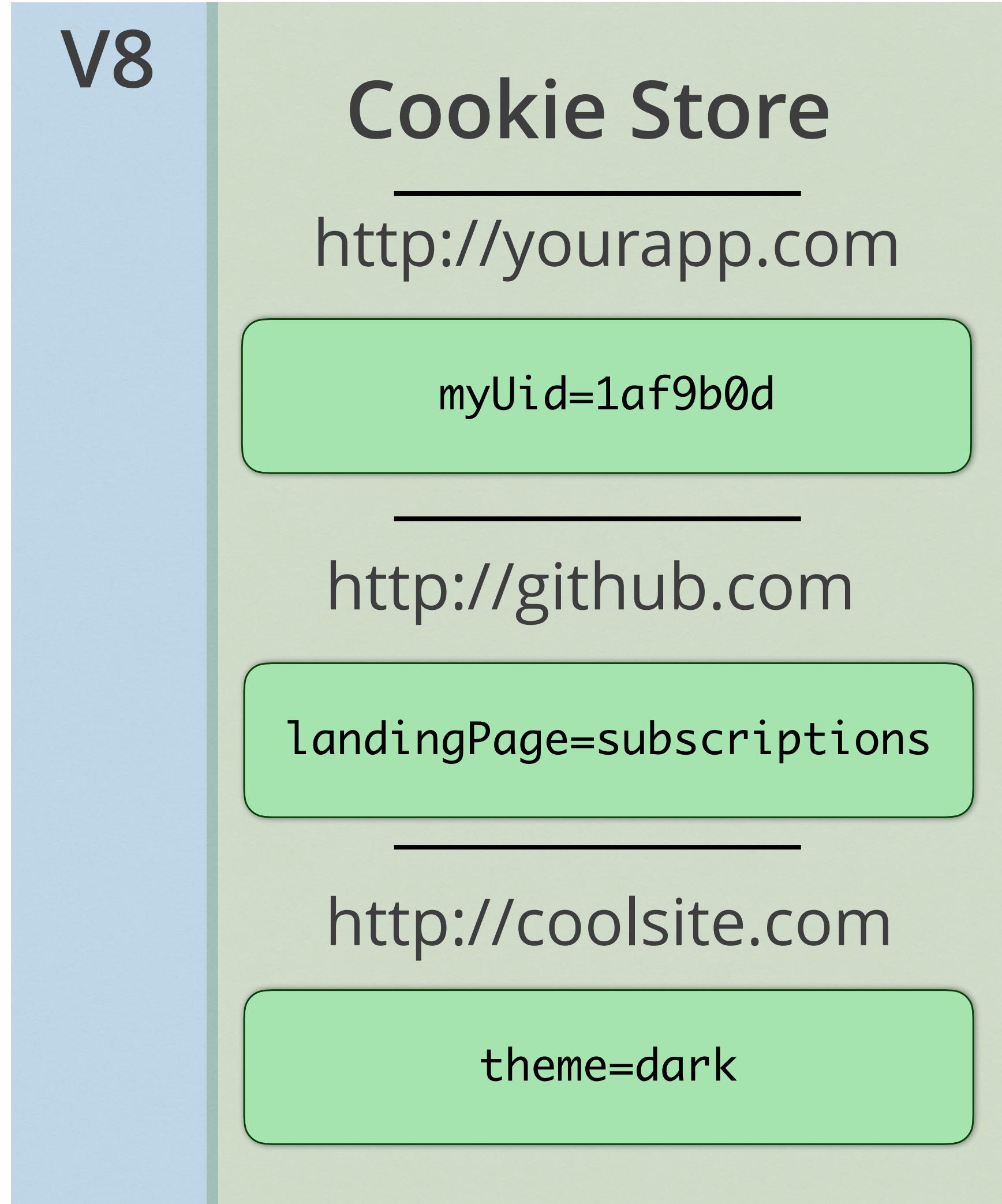


Internet (HTTP)

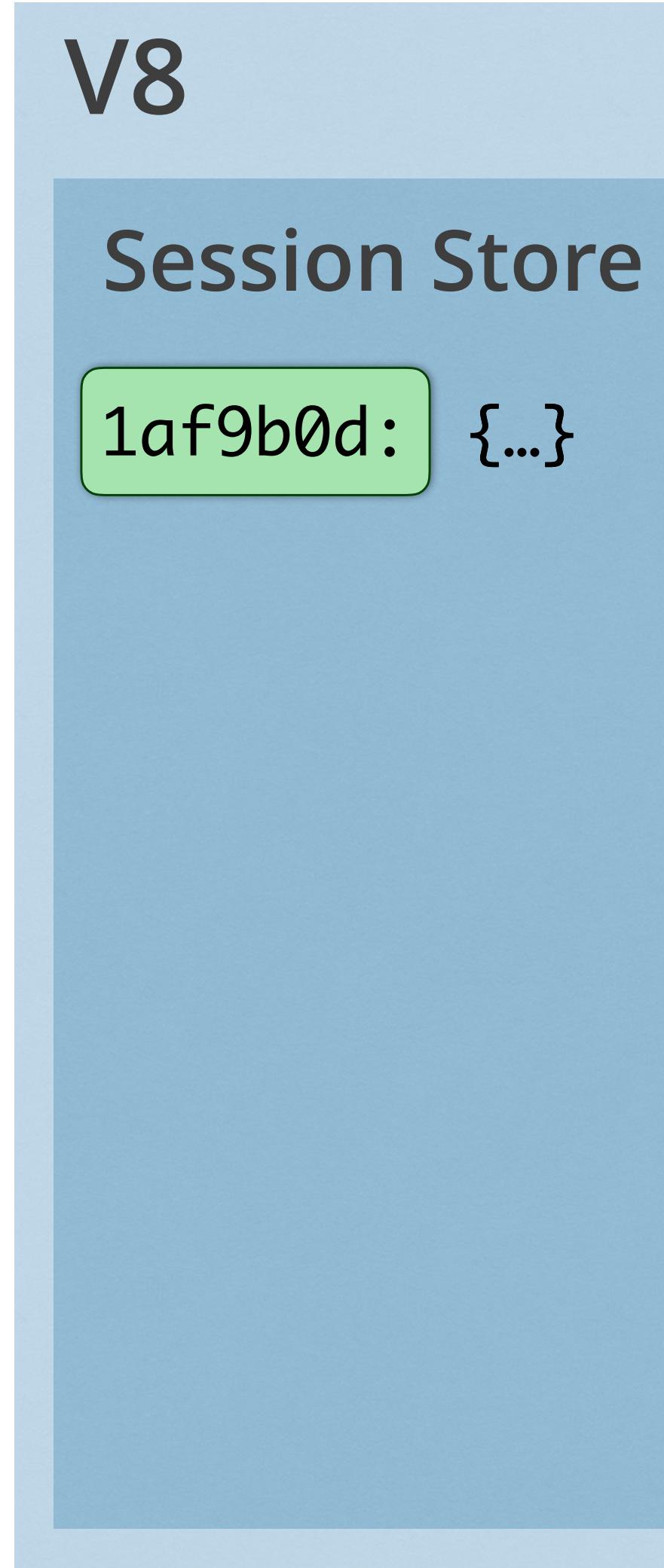


<http://yourapp.com>

Client (Browser)



Server (Backend)



<http://yourapp.com>

Internet (HTTP)

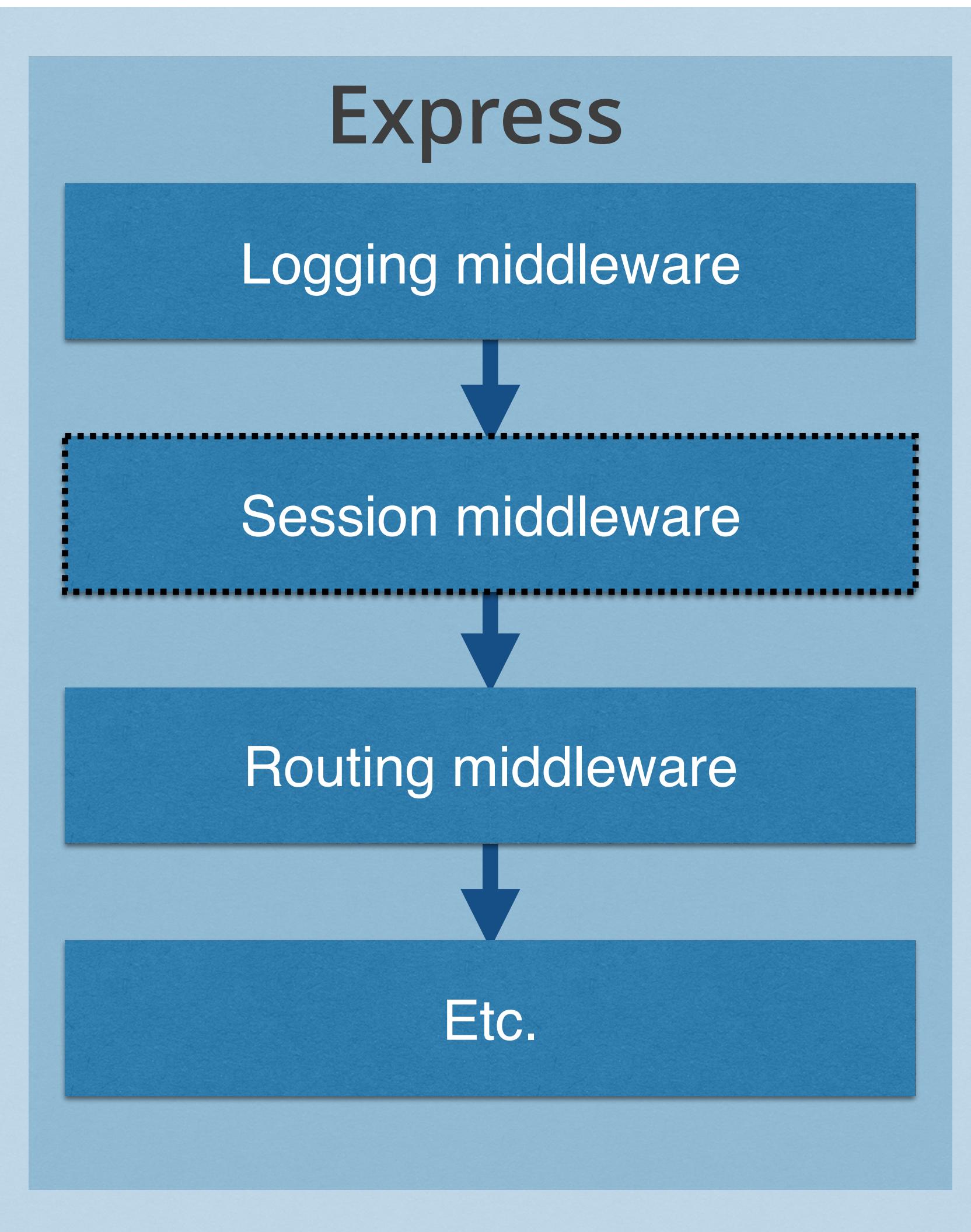


Multiple Sessions

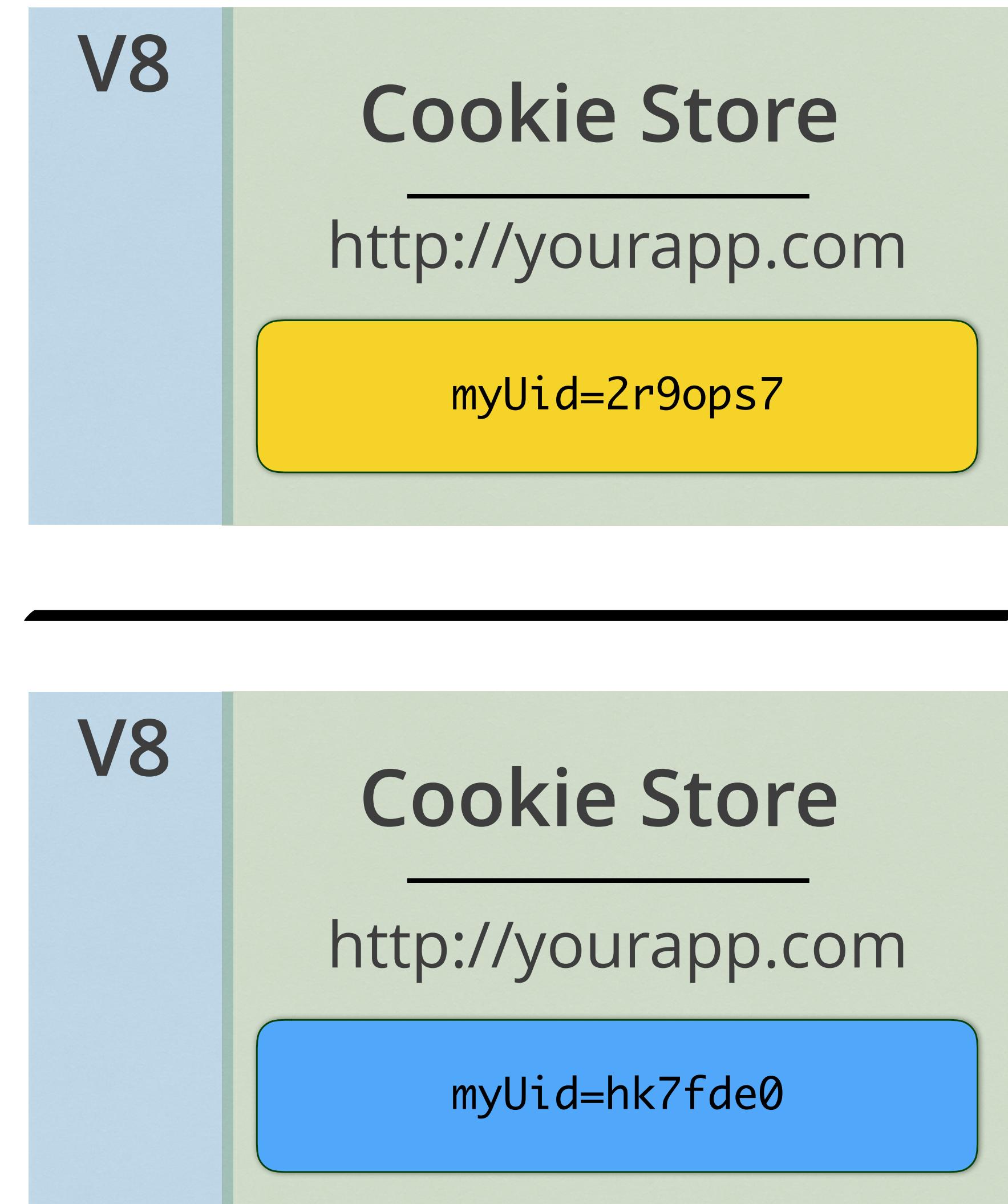
Server (Backend)



Internet (HTTP)



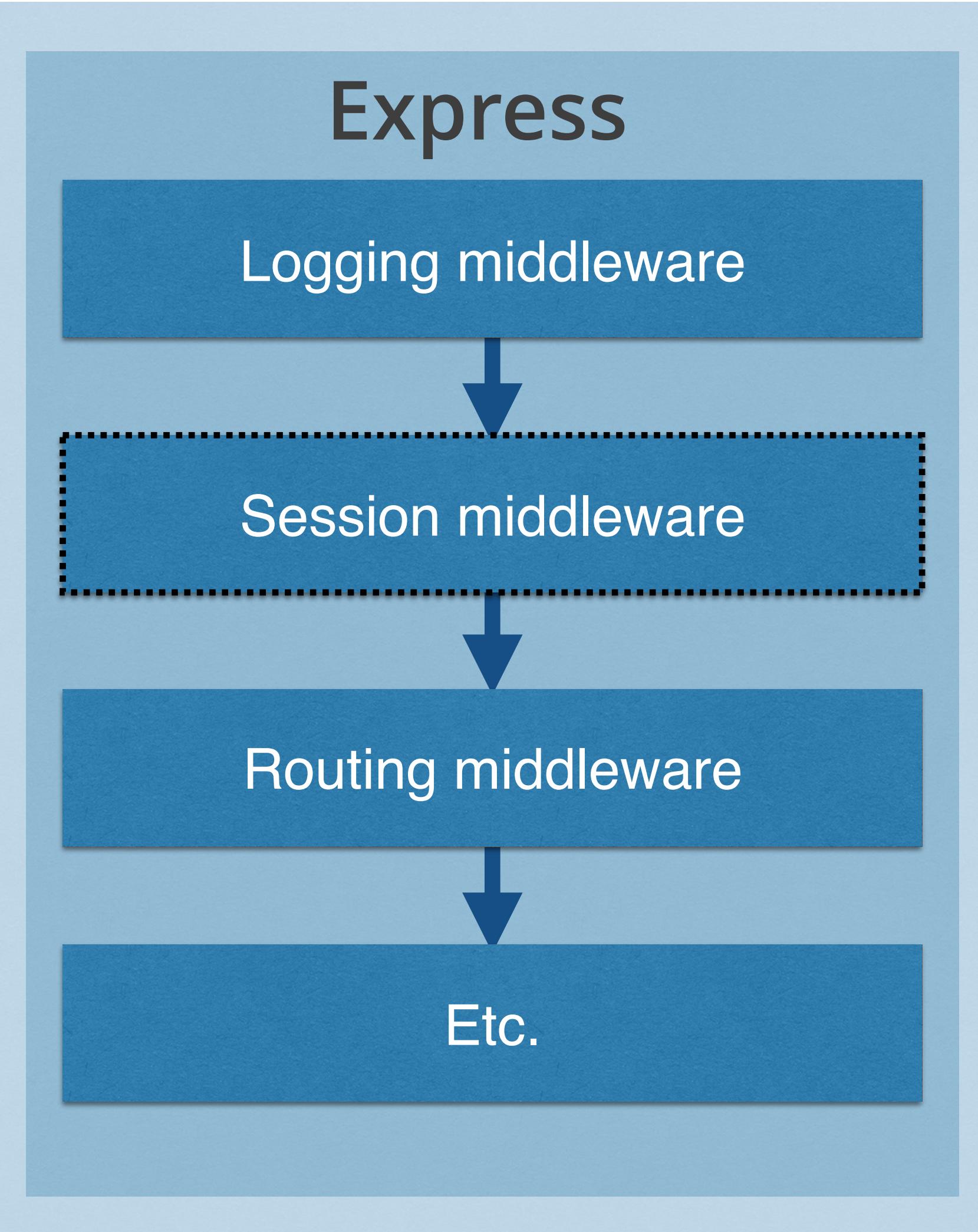
Client 1 (Browser)



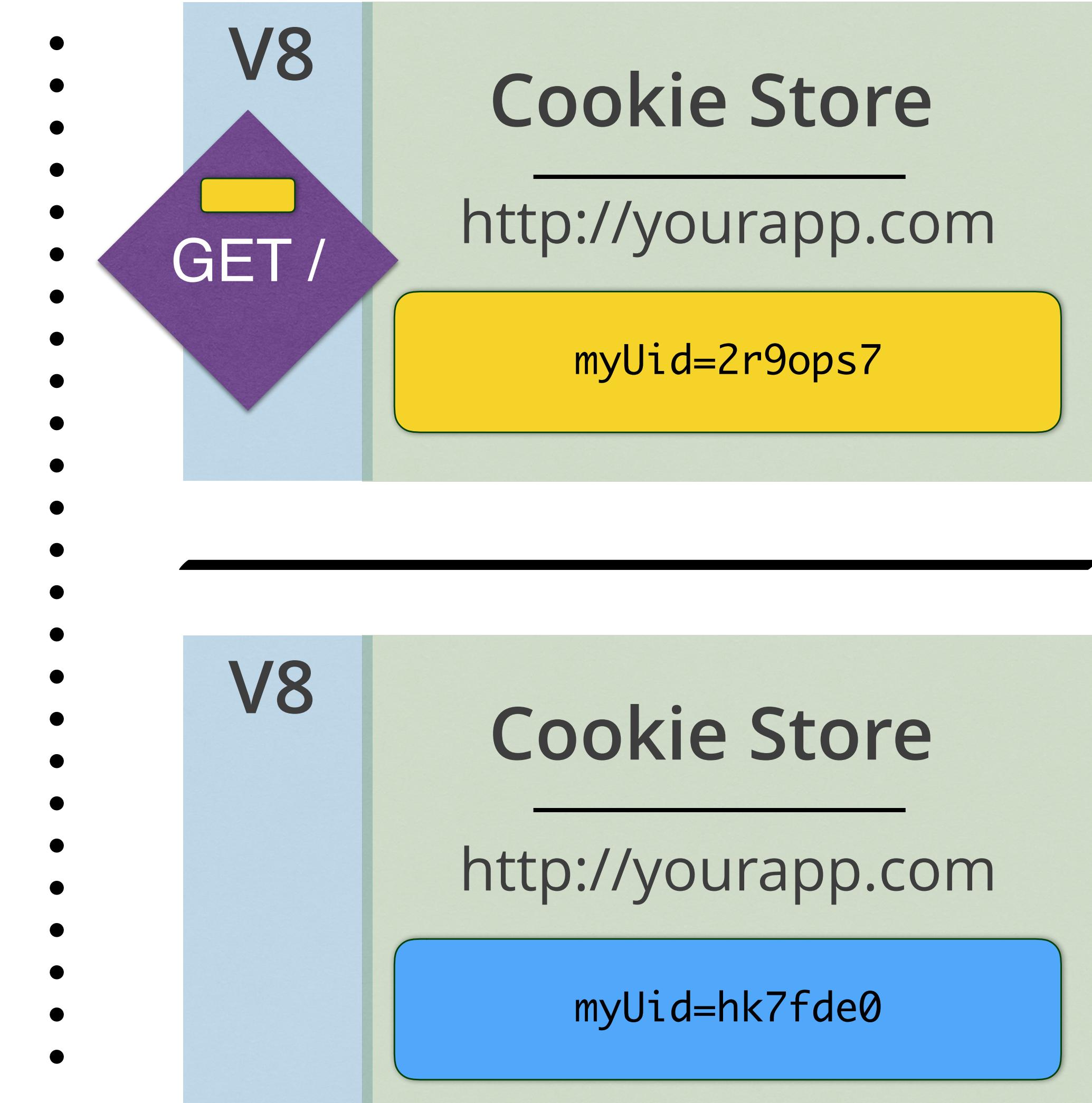
Server (Backend)



Internet (HTTP)



Client 1 (Browser)

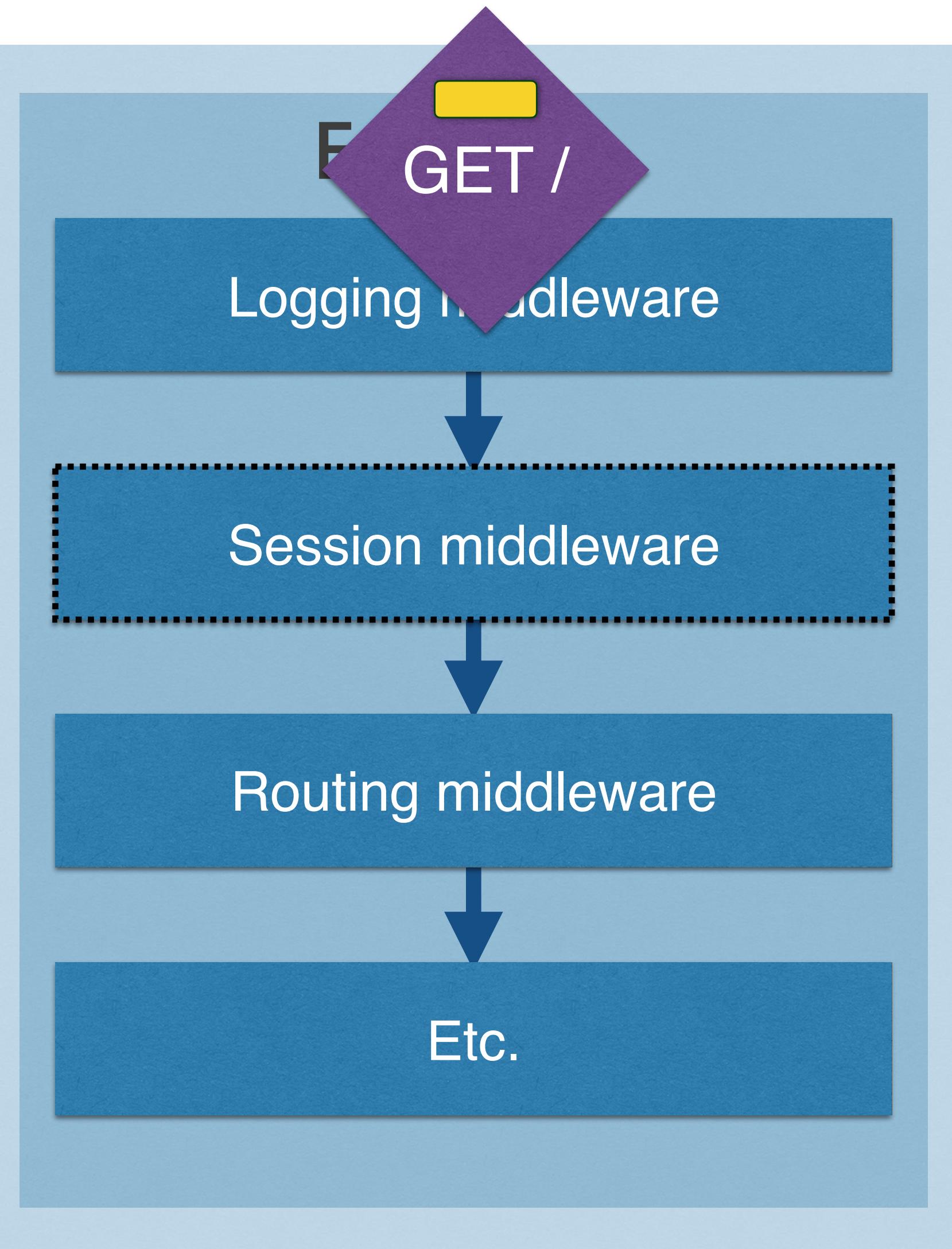


Client 2 (Browser)

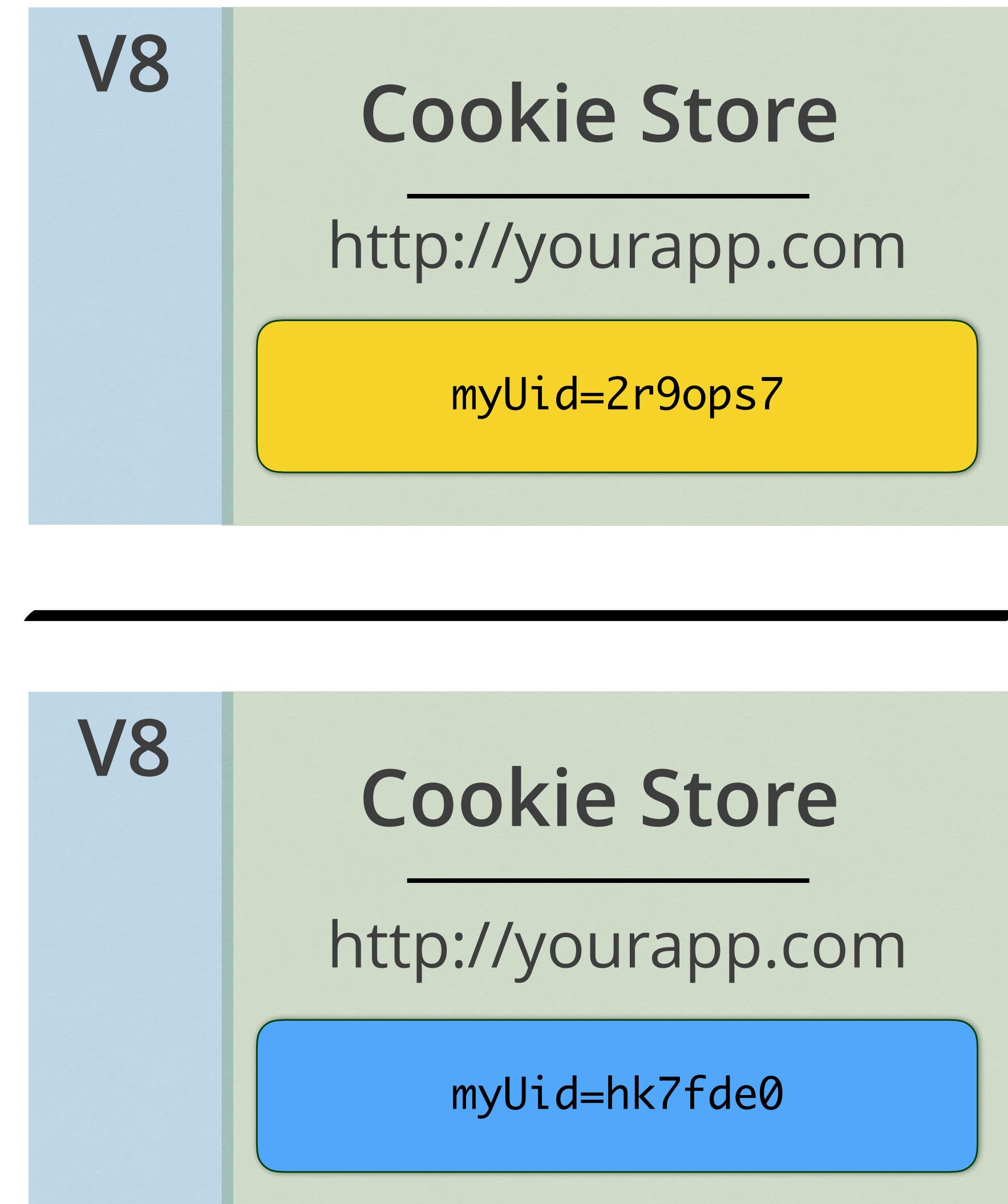
Server (Backend)



Internet (HTTP)



Client 1 (Browser)

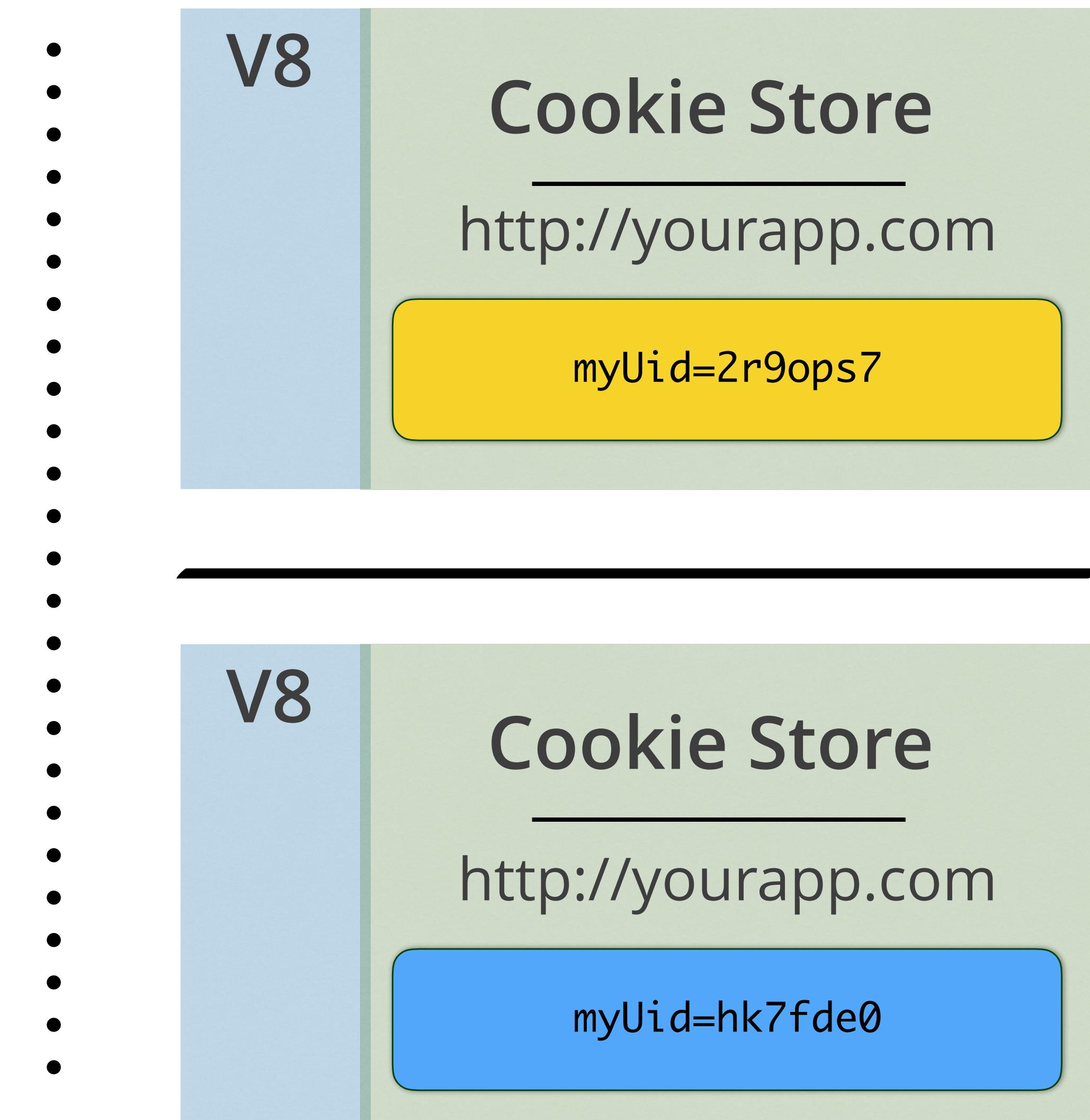


Server (Backend)



<http://yourapp.com>

Internet (HTTP)



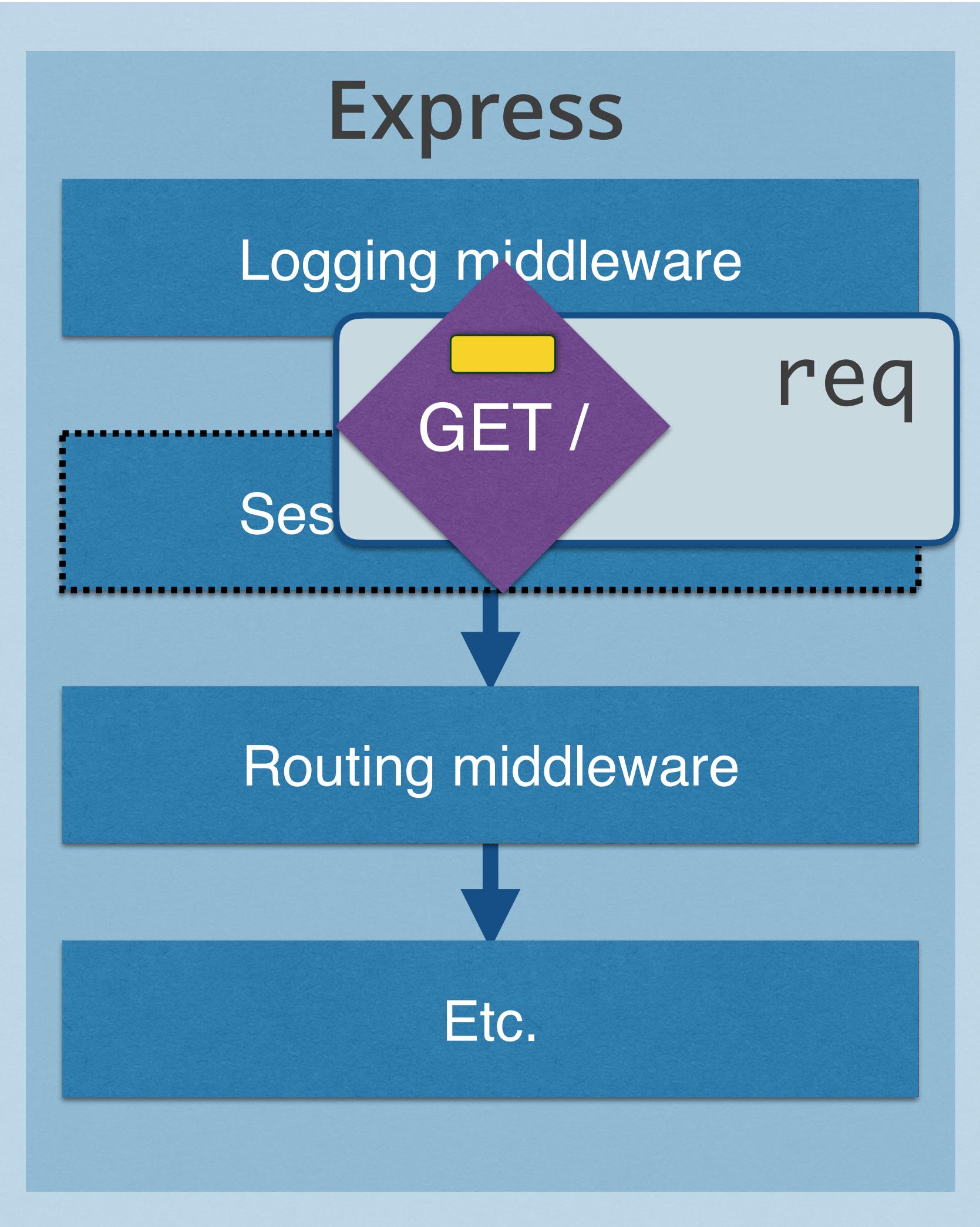
Client 2 (Browser)

Server (Backend)



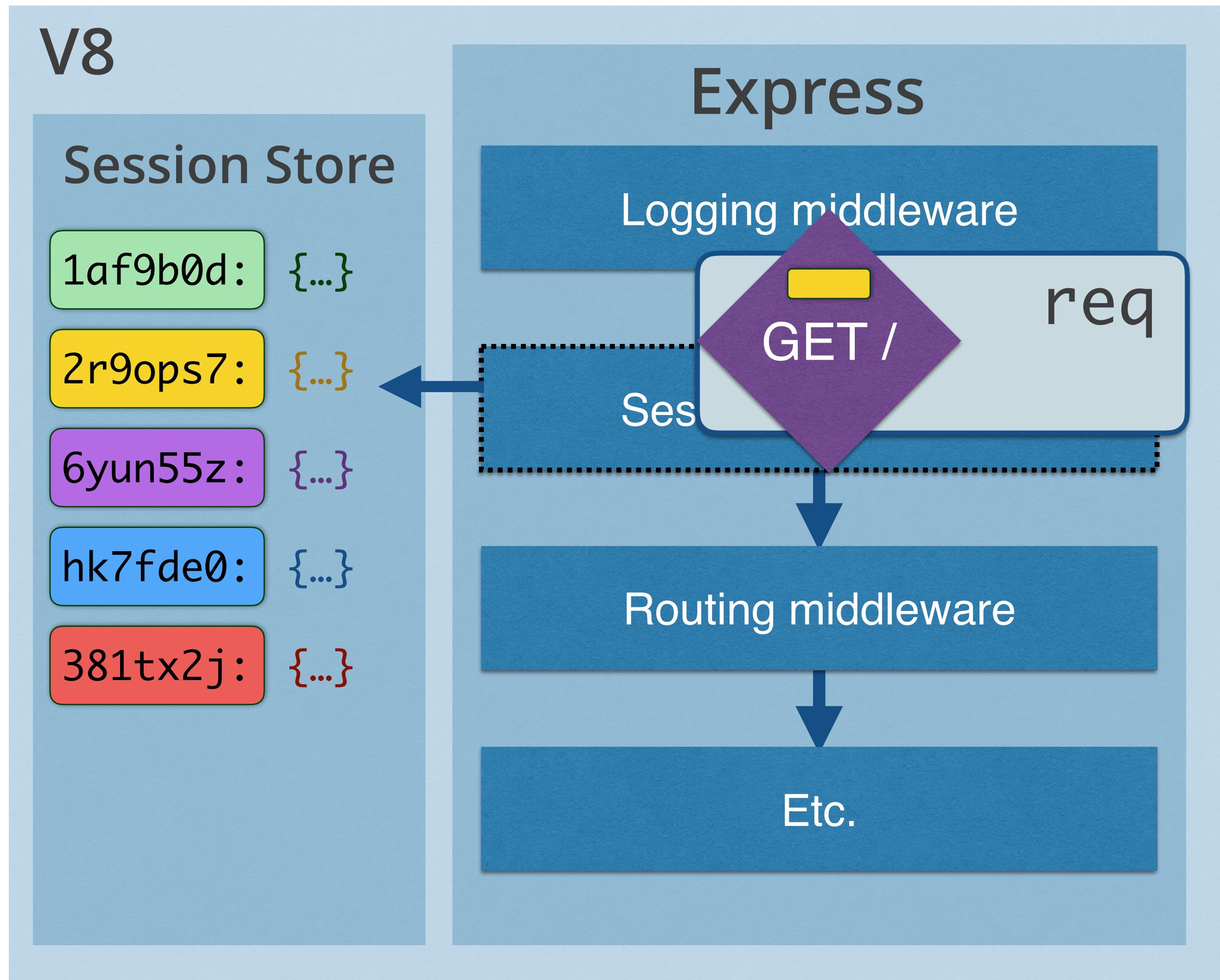
<http://yourapp.com>

Internet (HTTP)



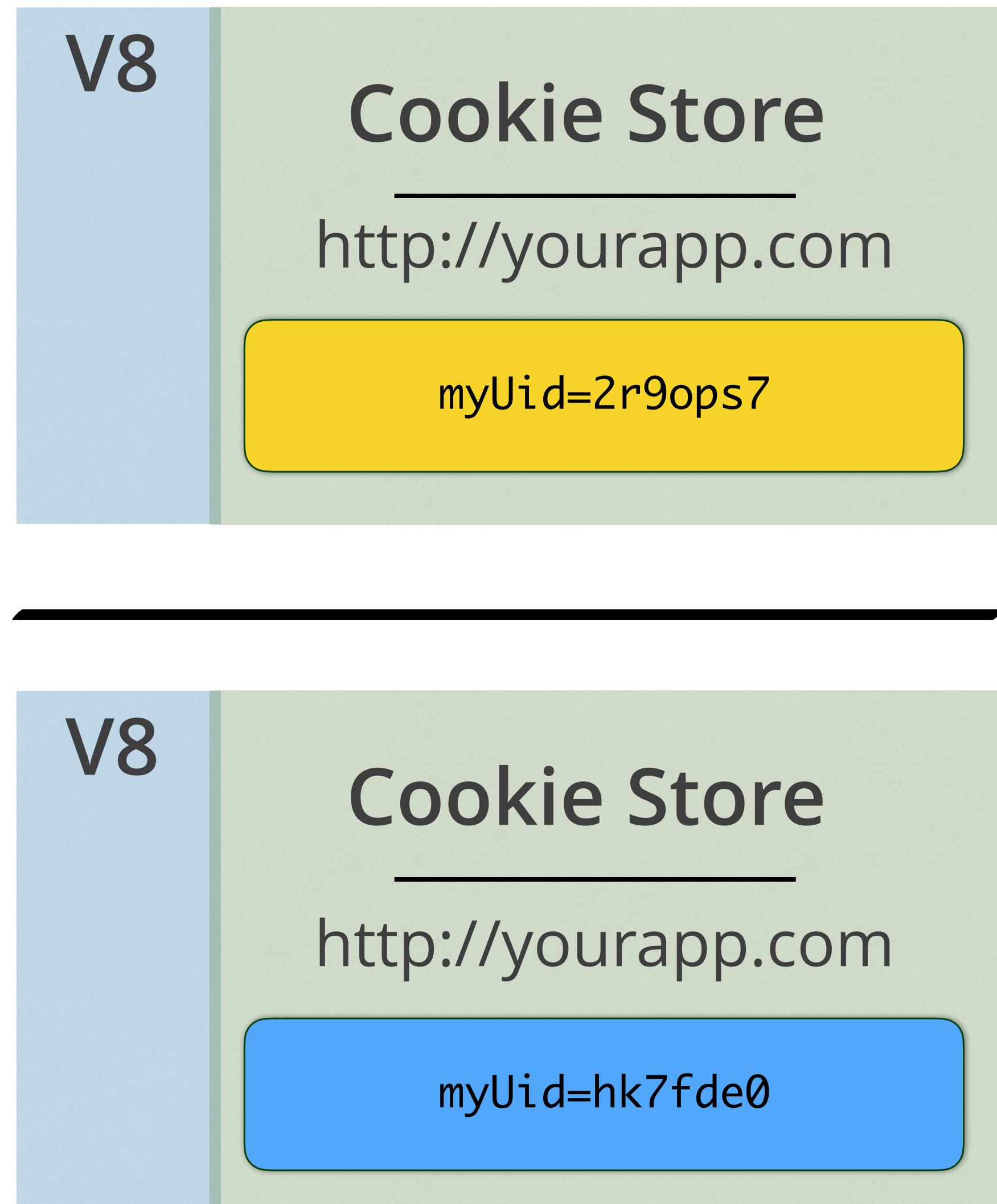
Client 2 (Browser)

Server (Backend)



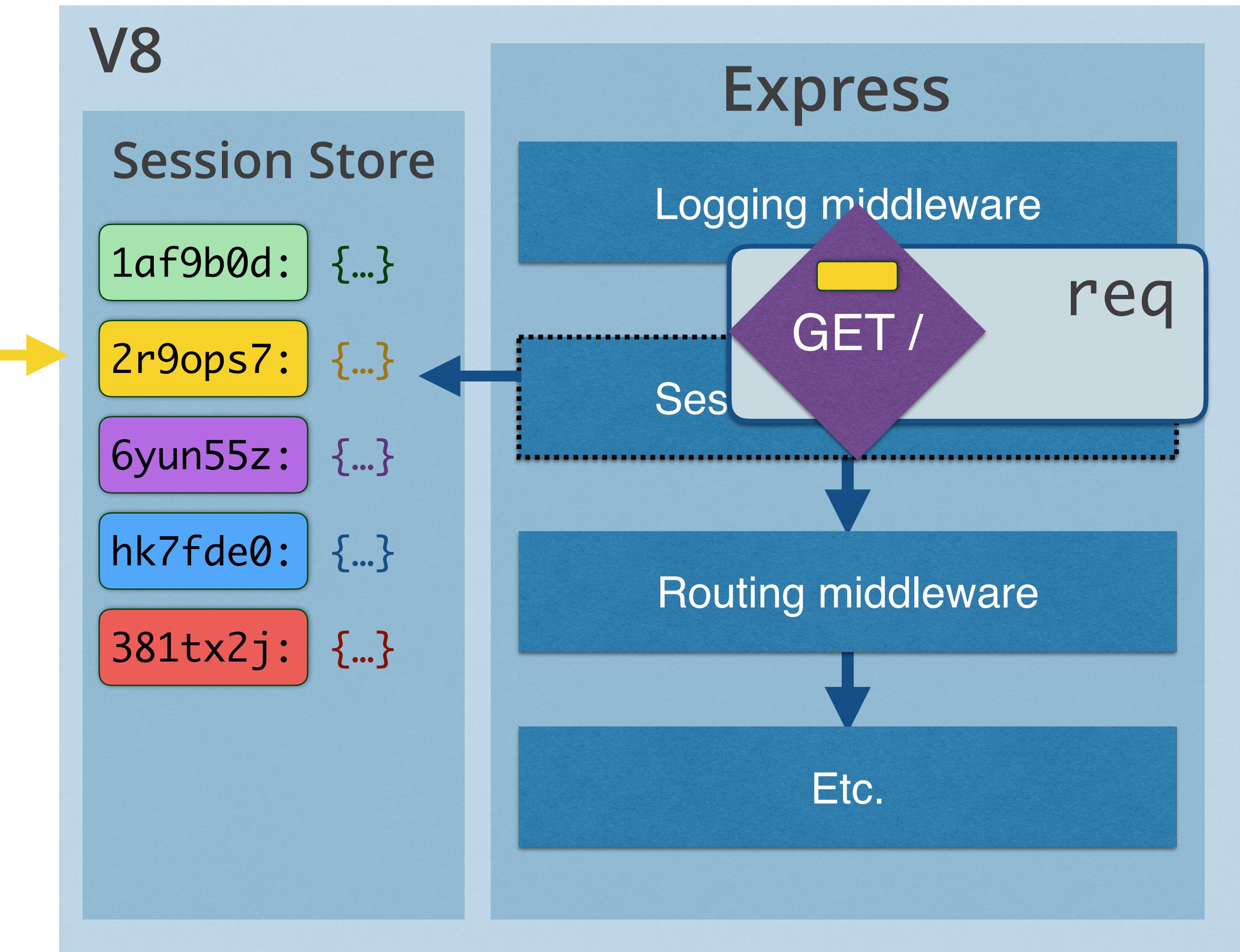
`http://yourapp.com`

Internet (HTTP)



`Client 2 (Browser)`

Server (Backend)



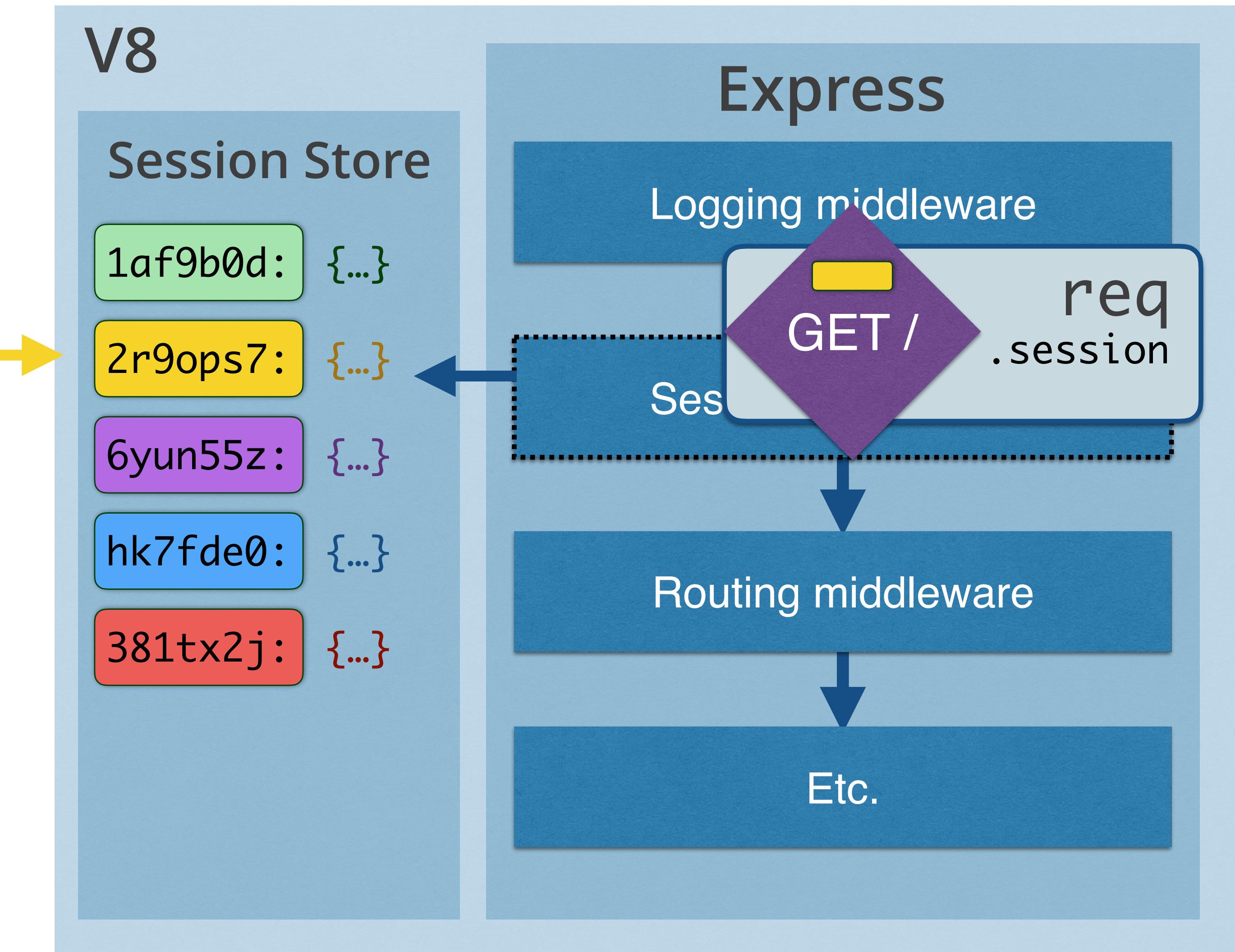
<http://yourapp.com>

Internet (HTTP)



Client 2 (Browser)

Server (Backend)



<http://yourapp.com>

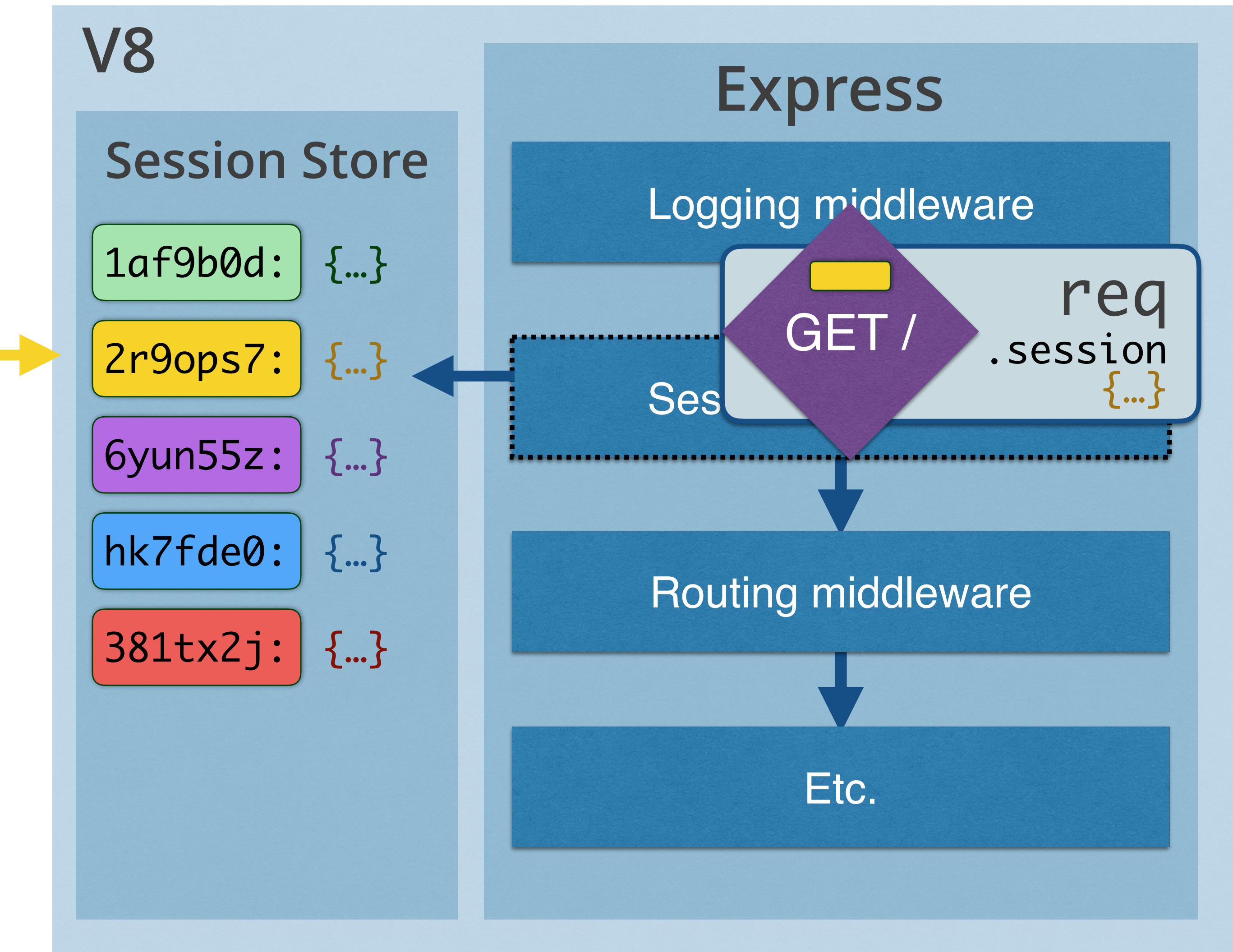
Internet (HTTP)

Client 1 (Browser)



Client 1 (Browser)

Server (Backend)



<http://yourapp.com>

Internet (HTTP)

Client 1 (Browser)

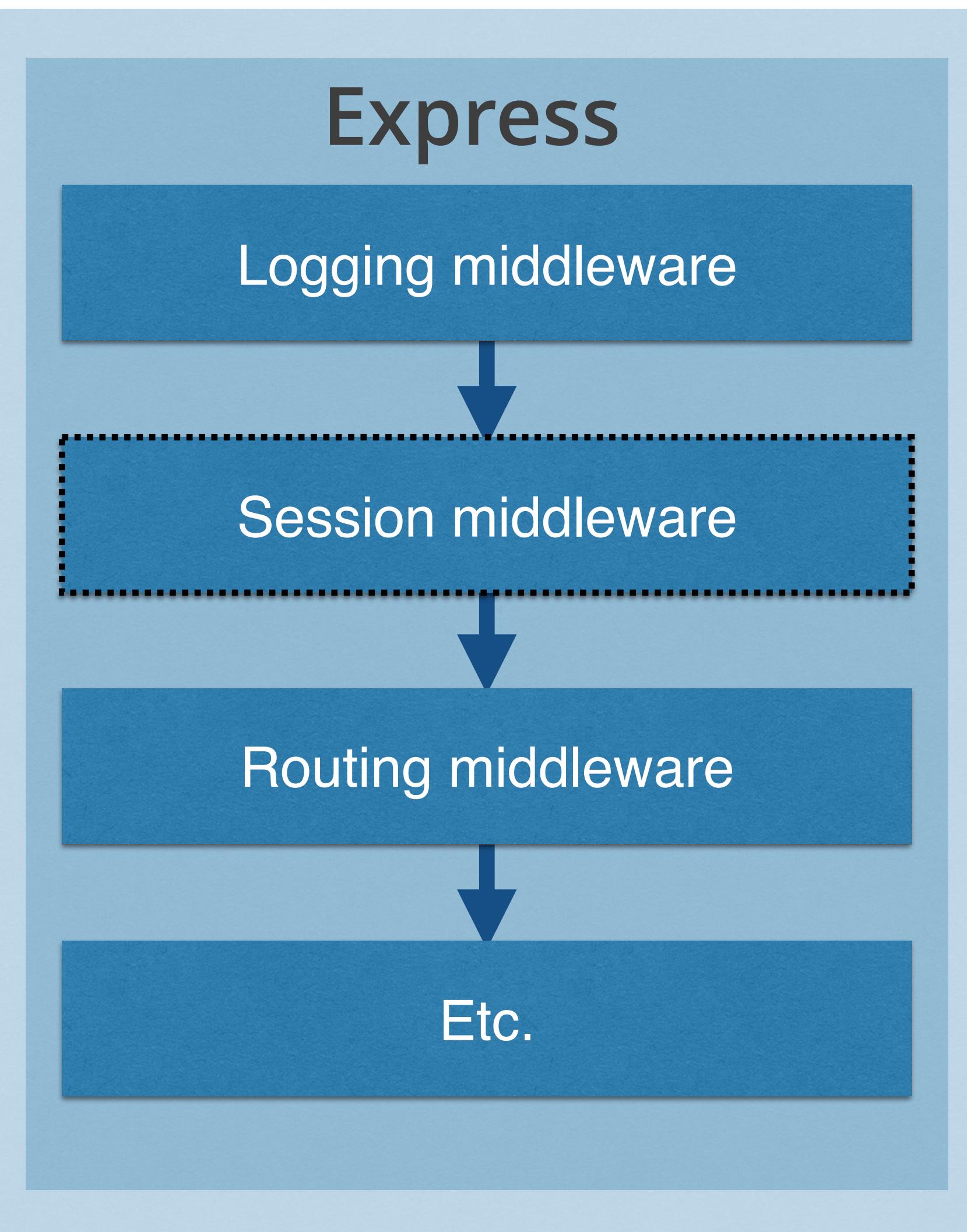


<http://yourapp.com>

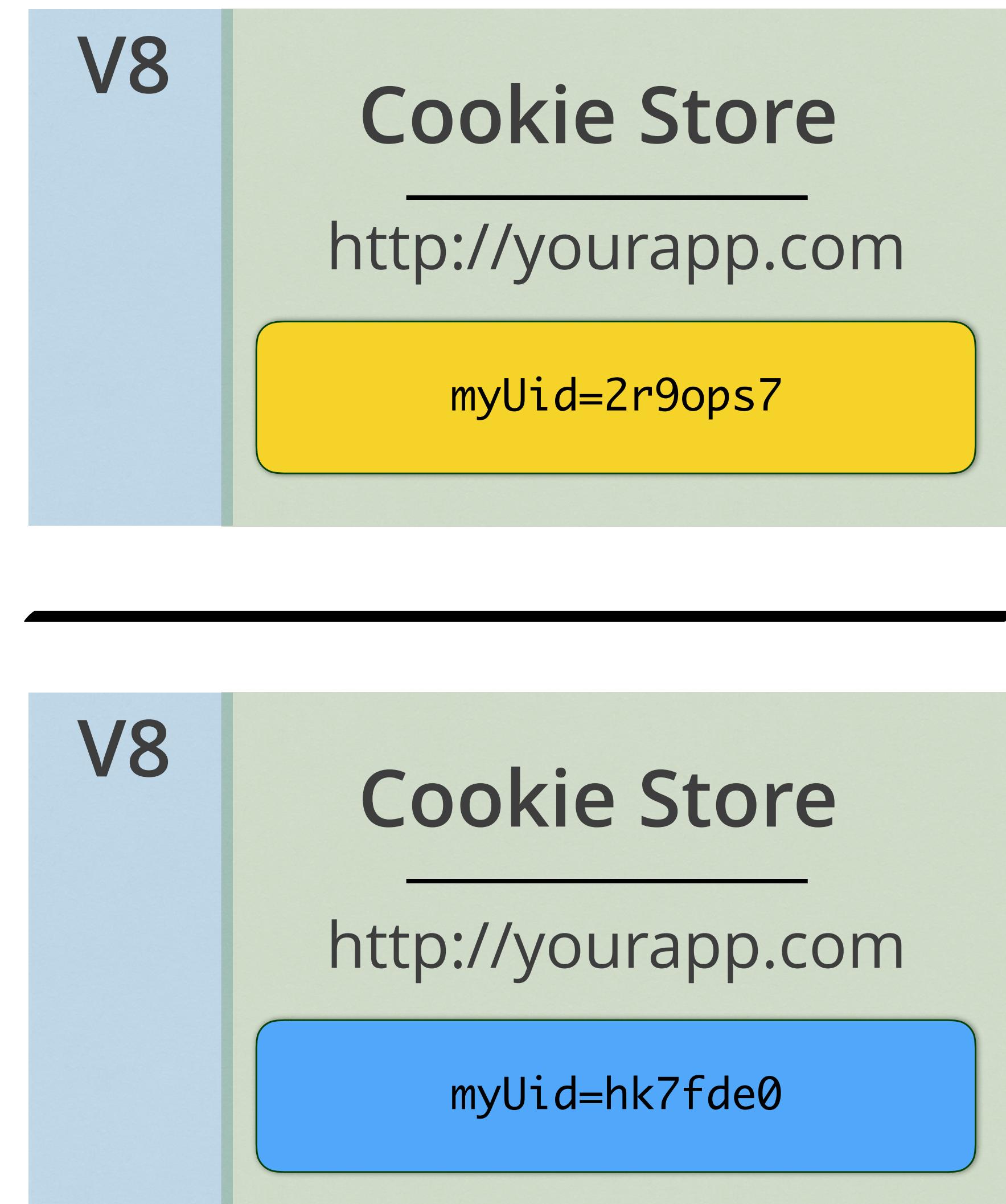
Server (Backend)



Internet (HTTP)



Client 1 (Browser)

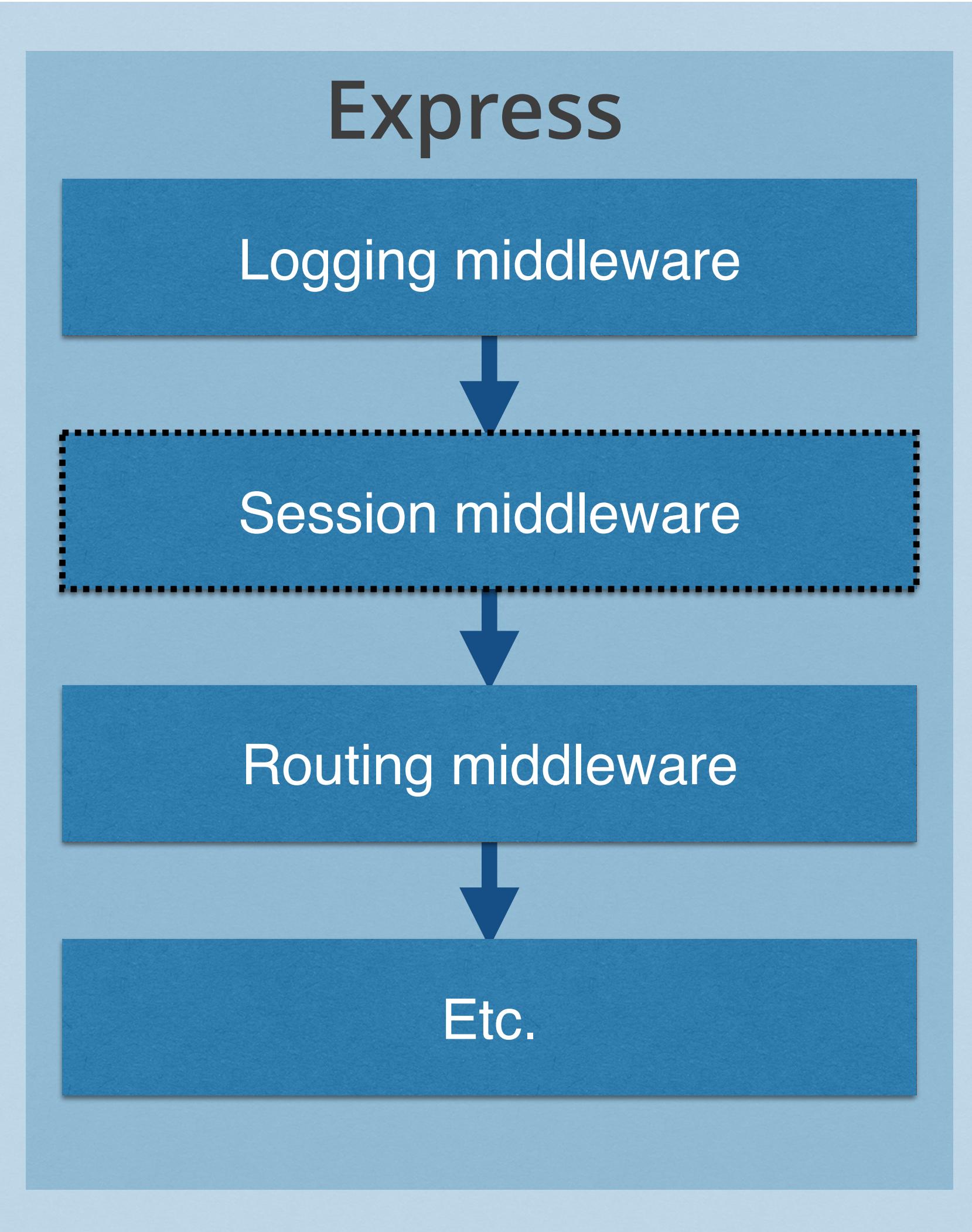


Client 2 (Browser)

Server (Backend)



Internet (HTTP)



Client 1 (Browser)



Client 2 (Browser)

Server (Backend)



Internet (HTTP)



Client 1 (Browser)

Cookie Store

http://yourapp.com

myUid=2r9ops7

Cookie Store

http://yourapp.com

myUid=hk7fde0

Client 2 (Browser)

Server (Backend)



<http://yourapp.com>

Internet (HTTP)



Client 2 (Browser)

Server (Backend)



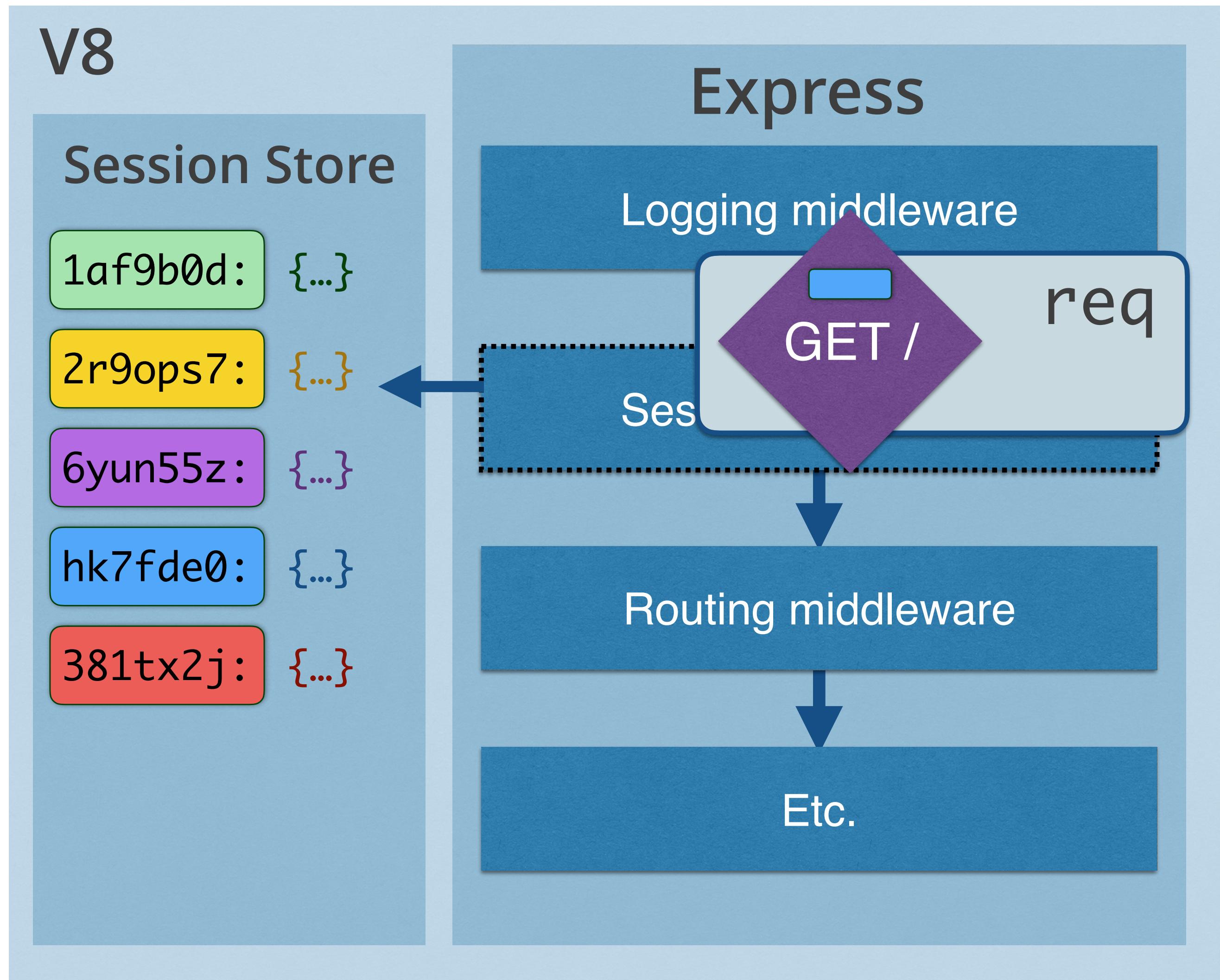
<http://yourapp.com>

Internet (HTTP)



Client 2 (Browser)

Server (Backend)



<http://yourapp.com>

Internet (HTTP)



Client 1 (Browser)

Cookie Store

<http://yourapp.com>

myUid=2r9ops7

V8

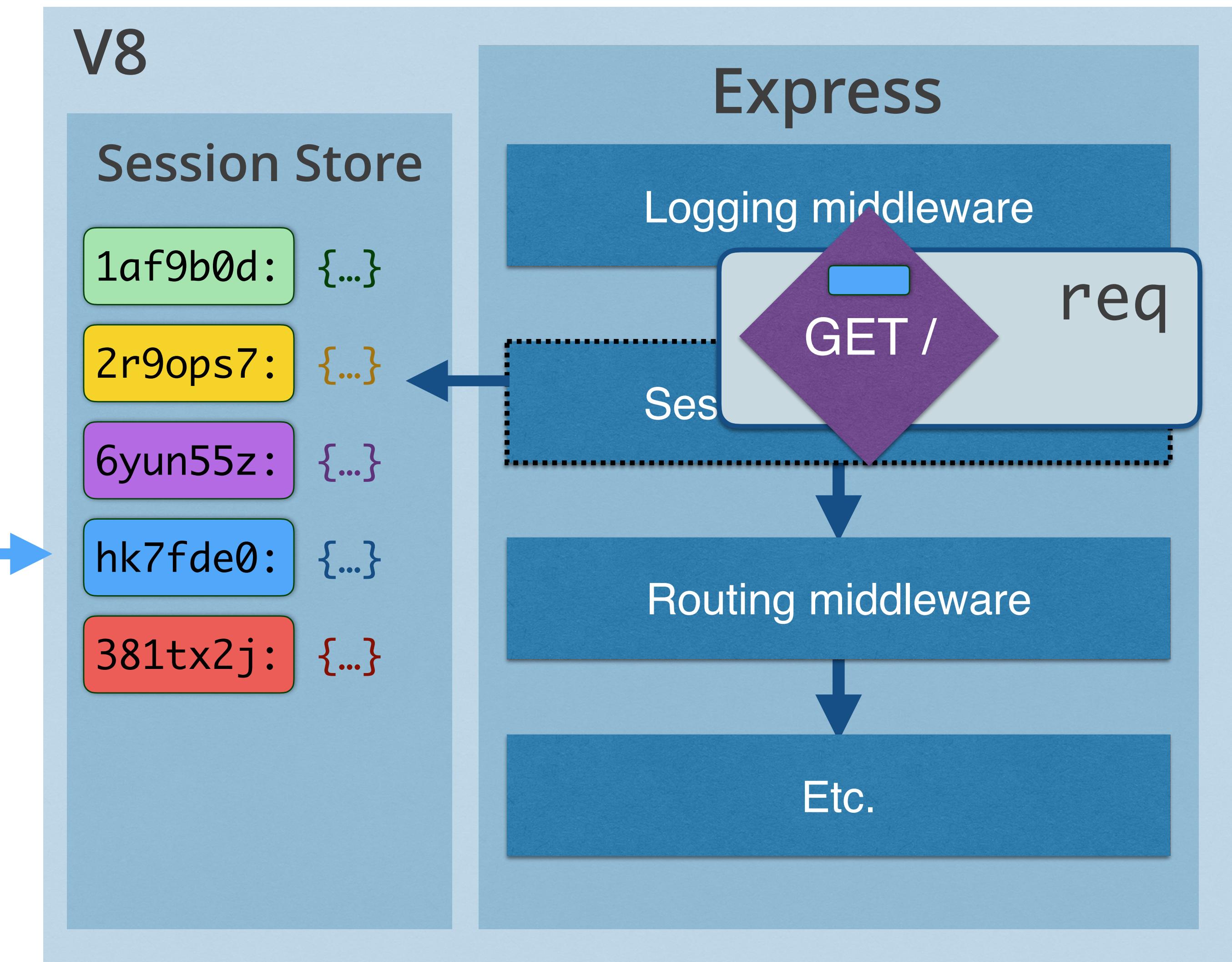
Cookie Store

<http://yourapp.com>

myUid=hk7fde0

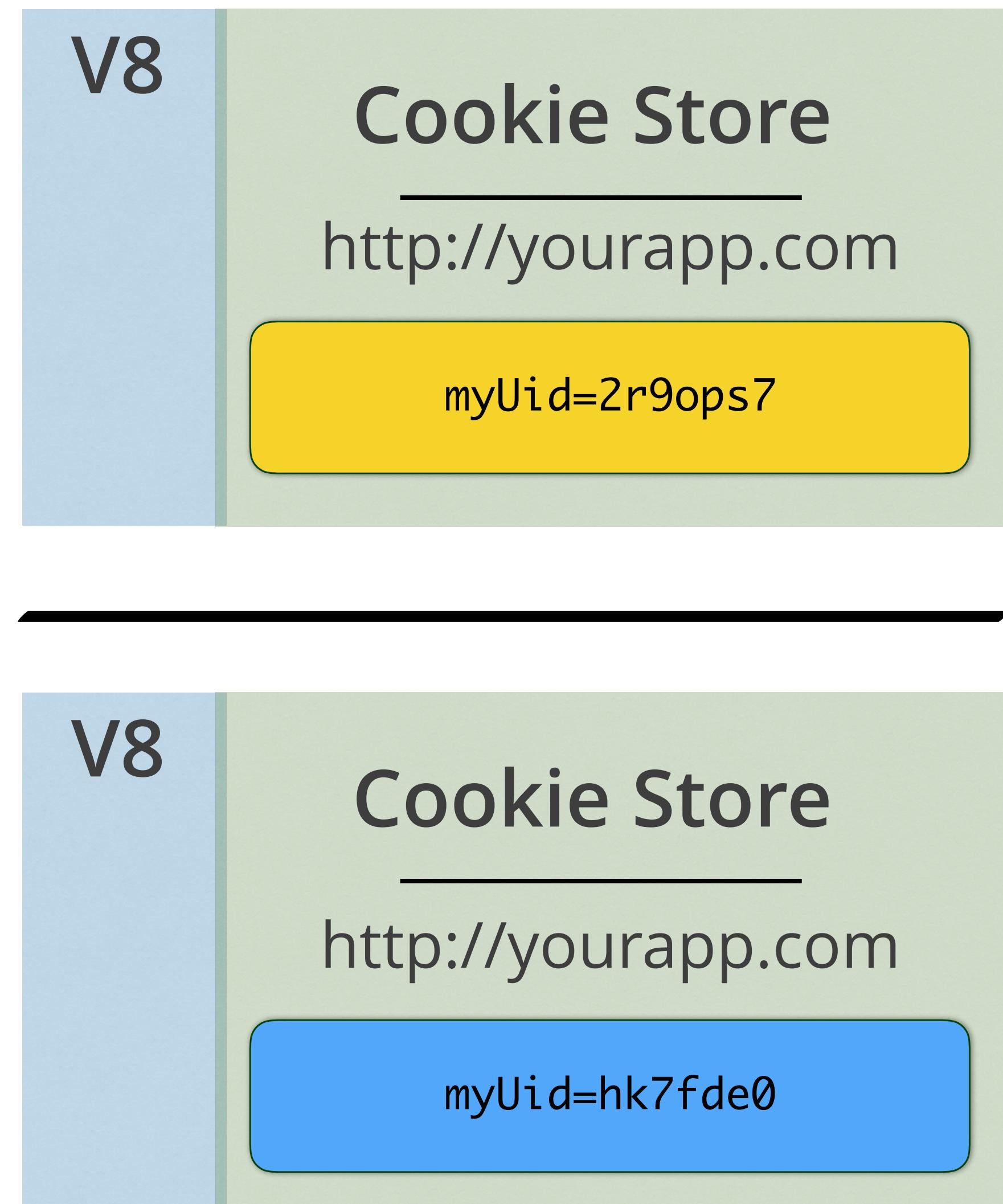
Client 2 (Browser)

Server (Backend)



<http://yourapp.com>

Internet (HTTP)

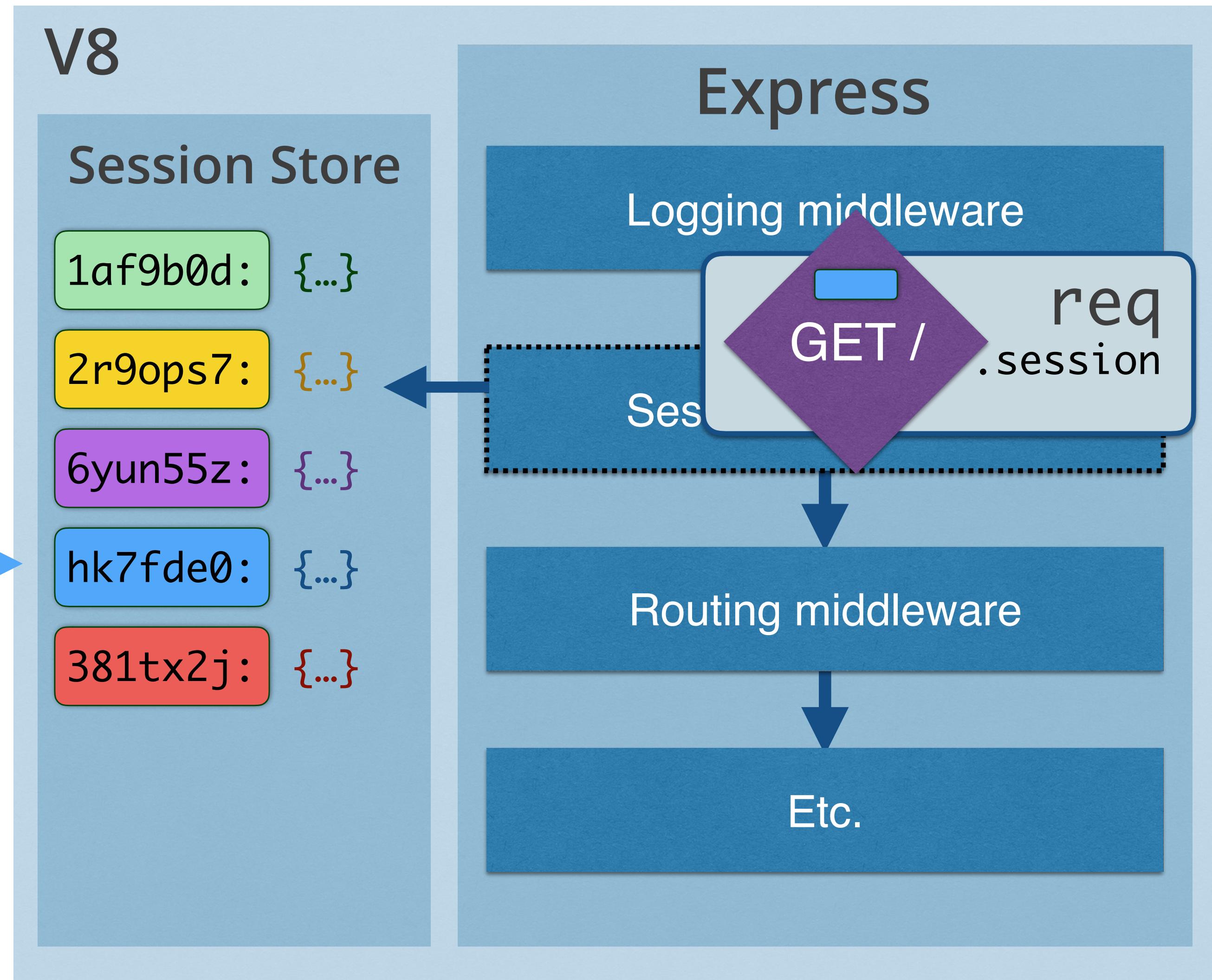


Client 2 (Browser)

Server (Backend)

Internet (HTTP)

Client 1 (Browser)



`http://yourapp.com`

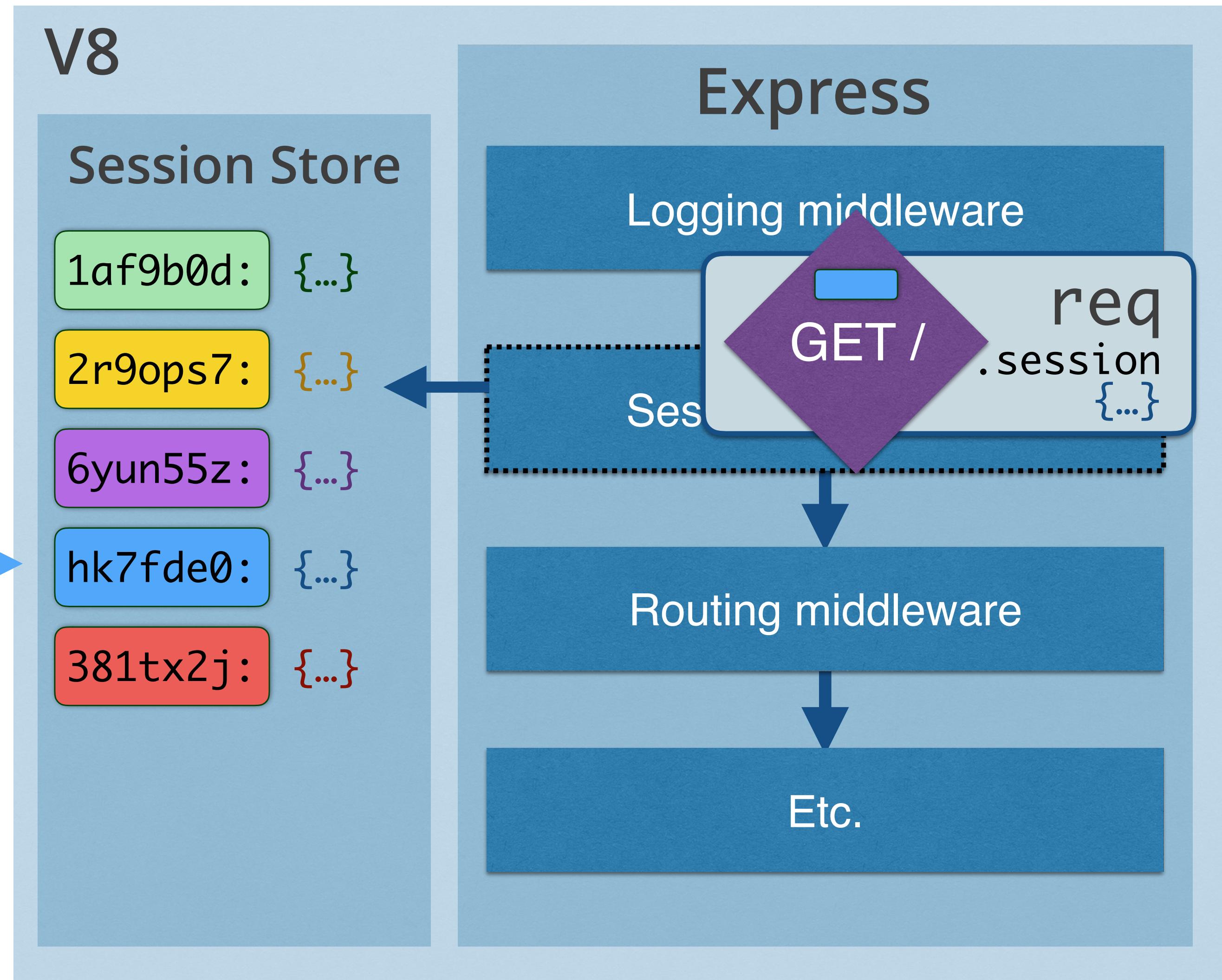


Client 2 (Browser)

Server (Backend)

Internet (HTTP)

Client 1 (Browser)



<http://yourapp.com>



Client 2 (Browser)

*“You get a session, you get a session,
everybody gets a session!”*

-NOT OPRAH





OAUTH

Standard protocol for auth piggybacking

Your Application

“consumer”

User

“user”

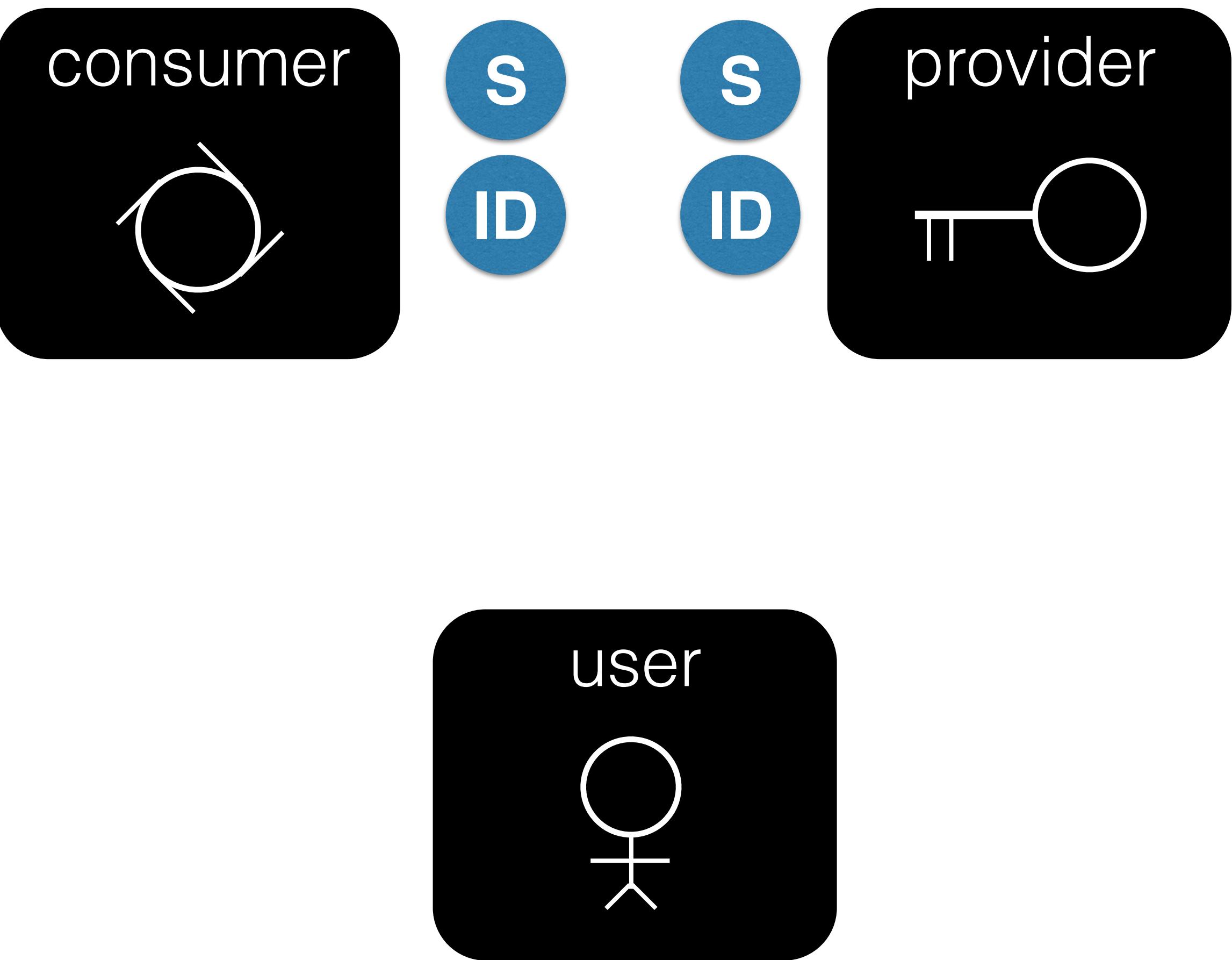
3rd Party Authority

“provider”



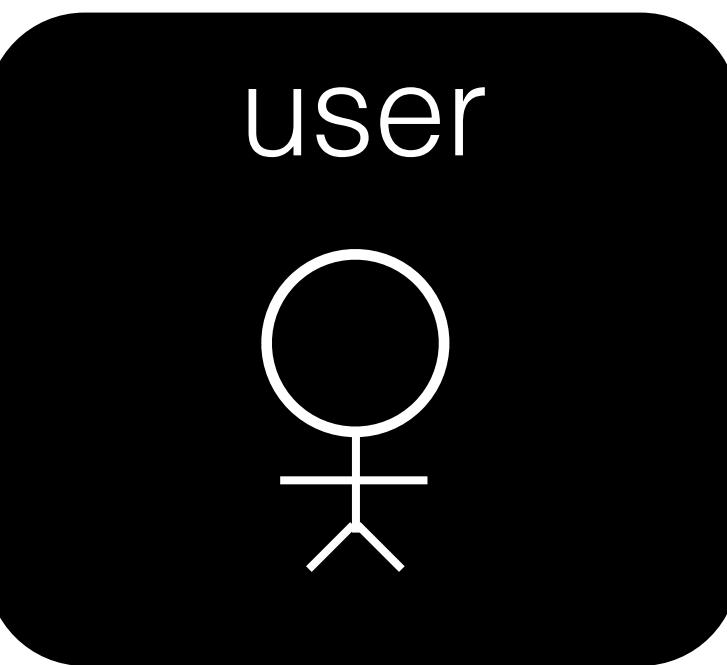
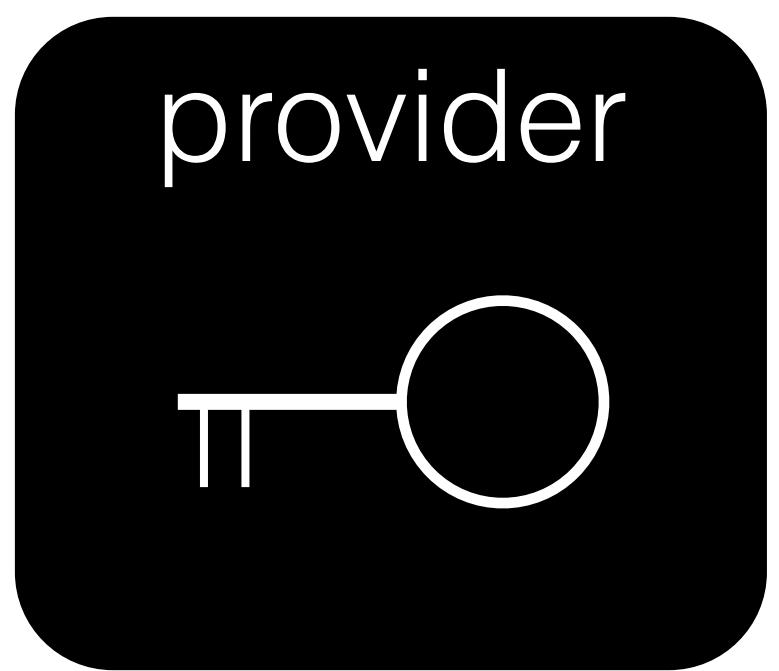
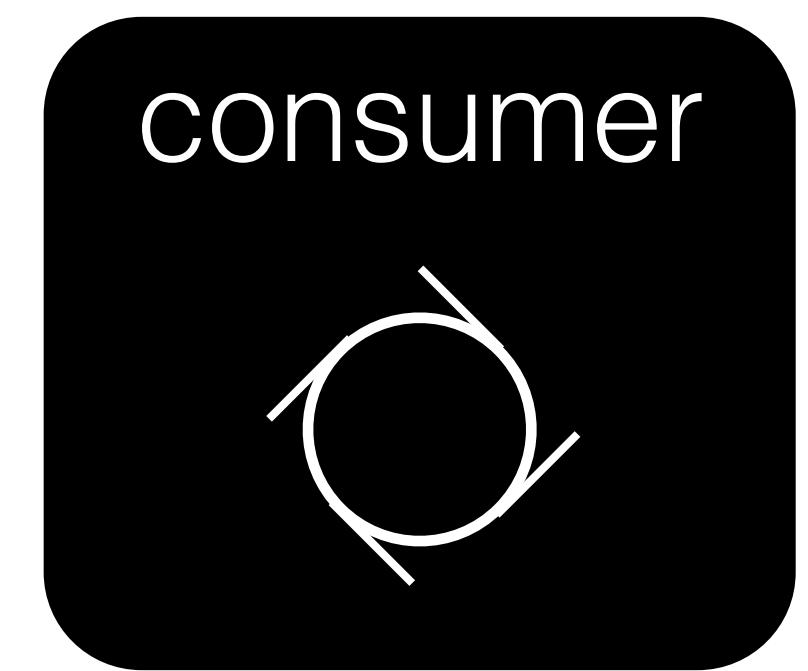
OAUTH - PREP

- Consumer app has a registered dev account with the provider.
- Provider gives consumer a public client ID and private client Secret.
- Both services hold on to these credentials so the consumer can prove who it is to the provider.





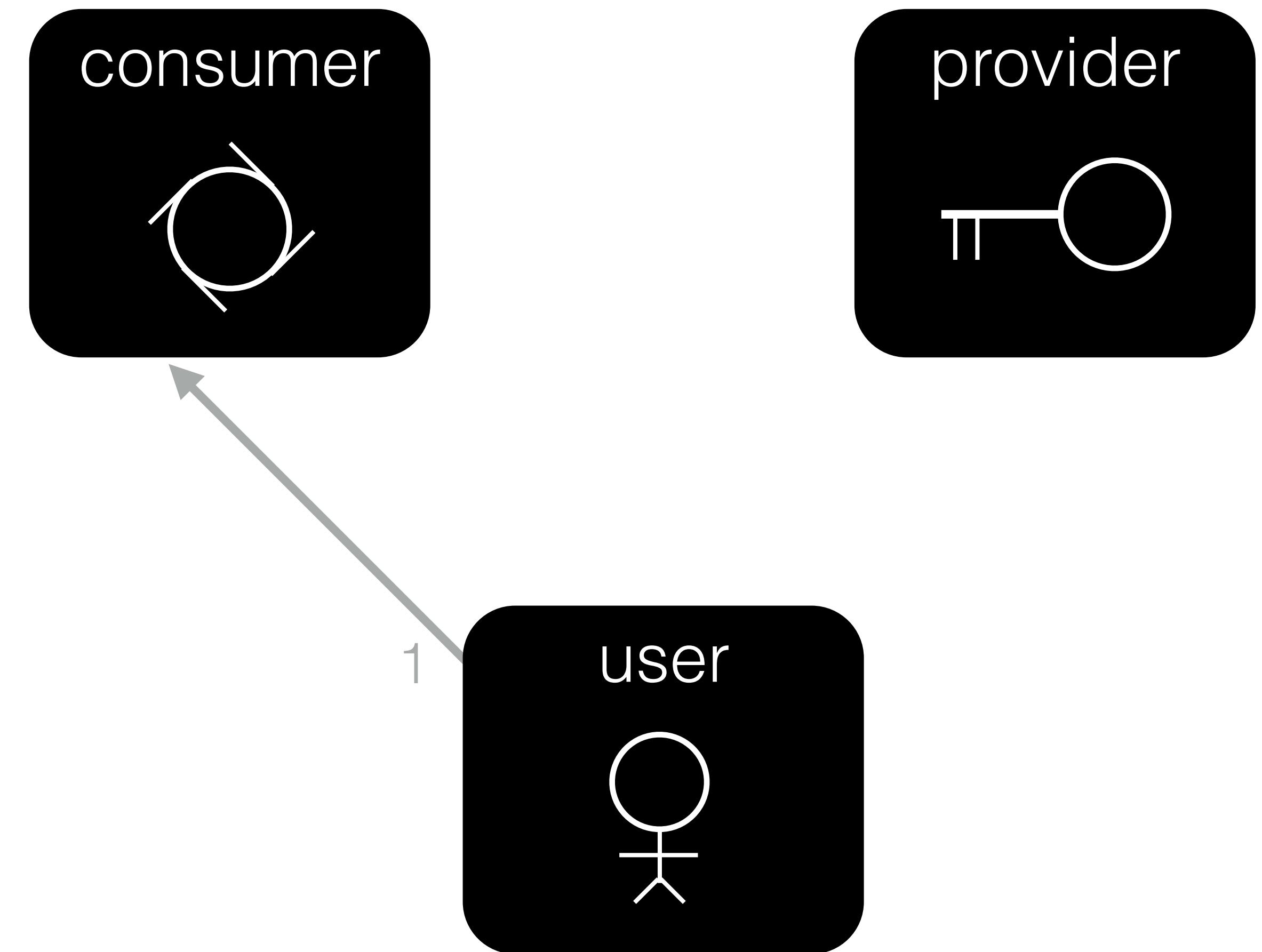
OAUTH - PETITION





OAUTH - PETITION

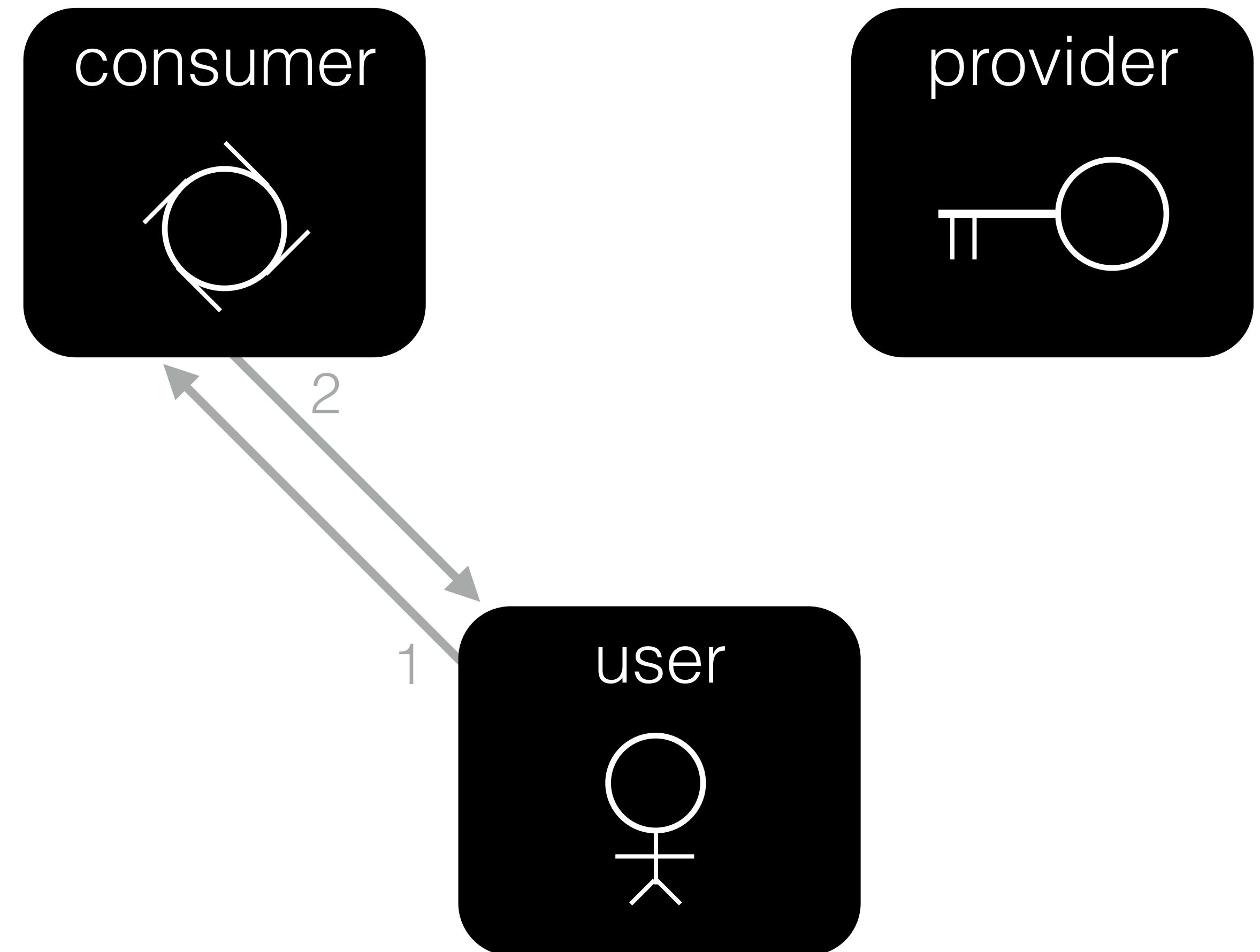
1. Request: load login





OAUTH - PETITION

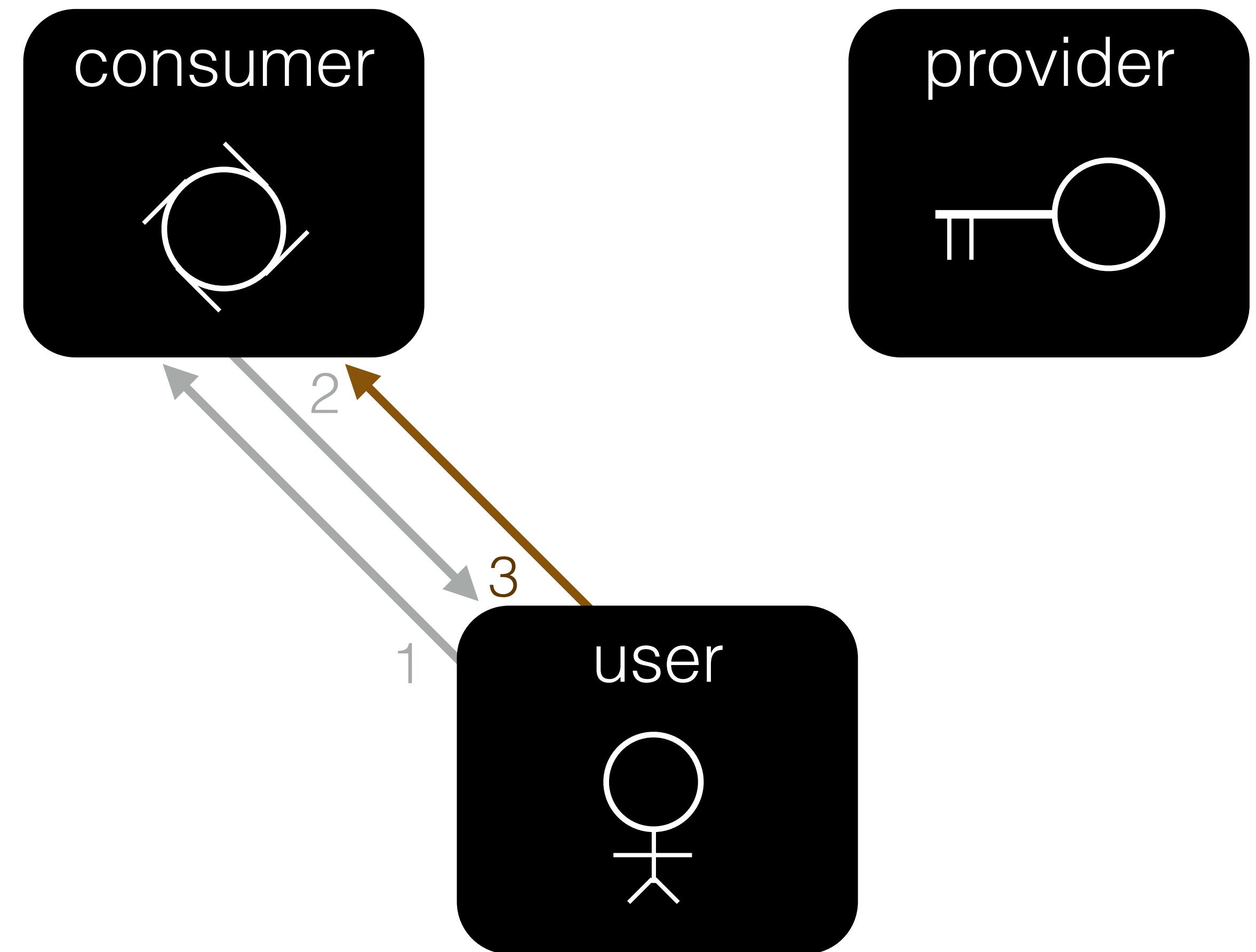
- 1.Request: load login
- 2.Response: rendered login





OAUTH - PETITION

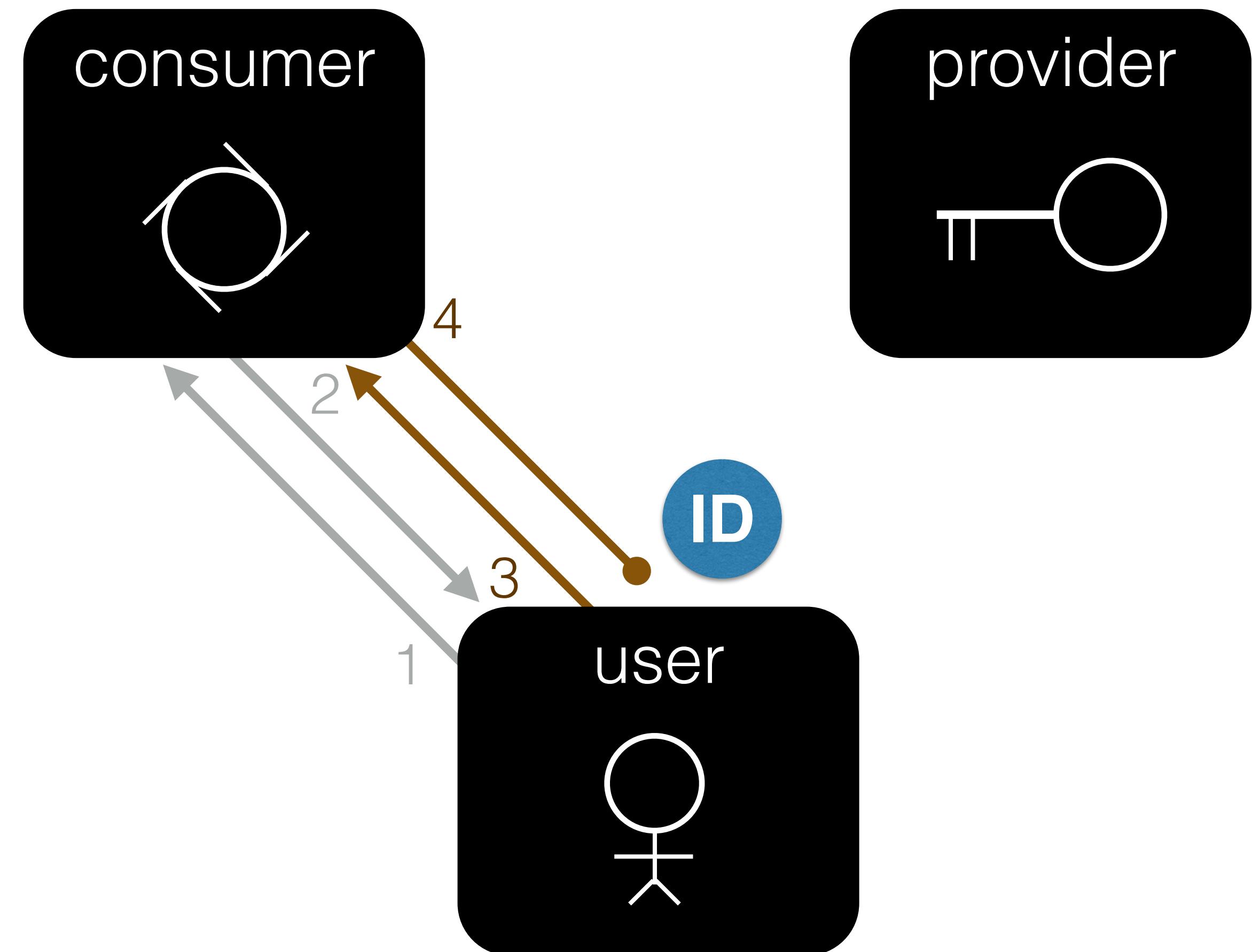
1. Request: load login
2. Response: rendered login
3. Request: login through provider





OAUTH - PETITION

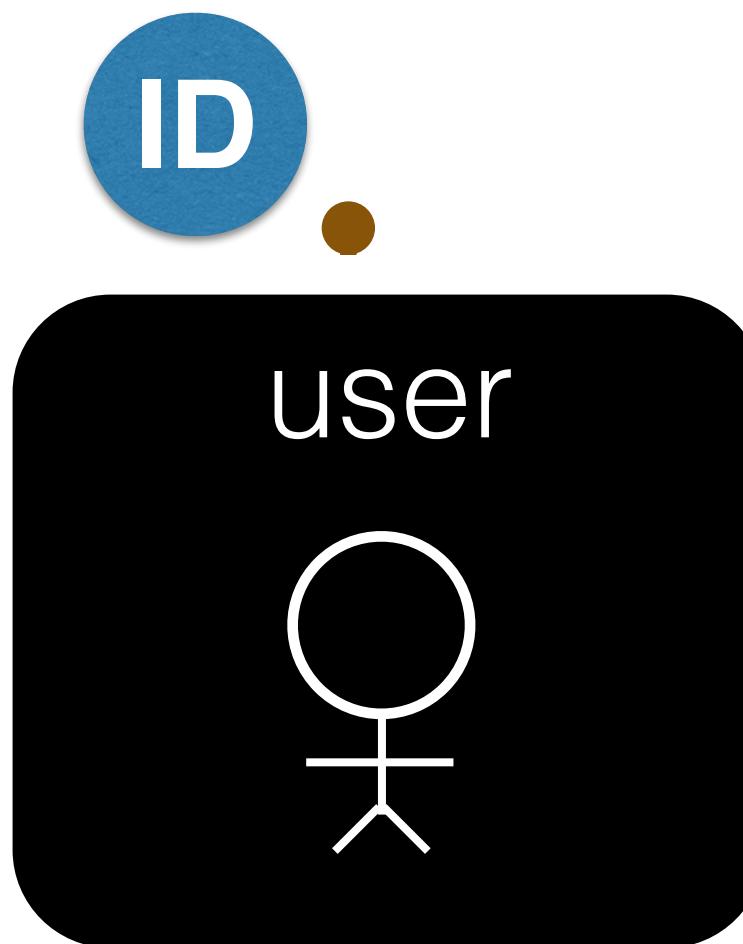
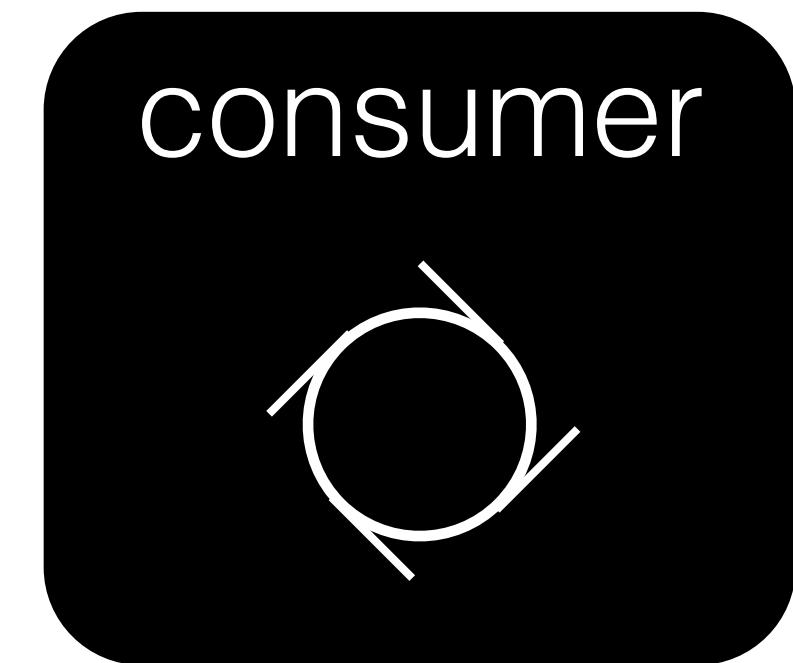
1. Request: load login
2. Response: rendered login
3. Request: login through provider
4. Response: redirect to provider



http://provider.com/oauth/authorize?client_id=123&callback_url=consumer.com/confirm&scope=read



OAUTH - USER AUTHENTICATION

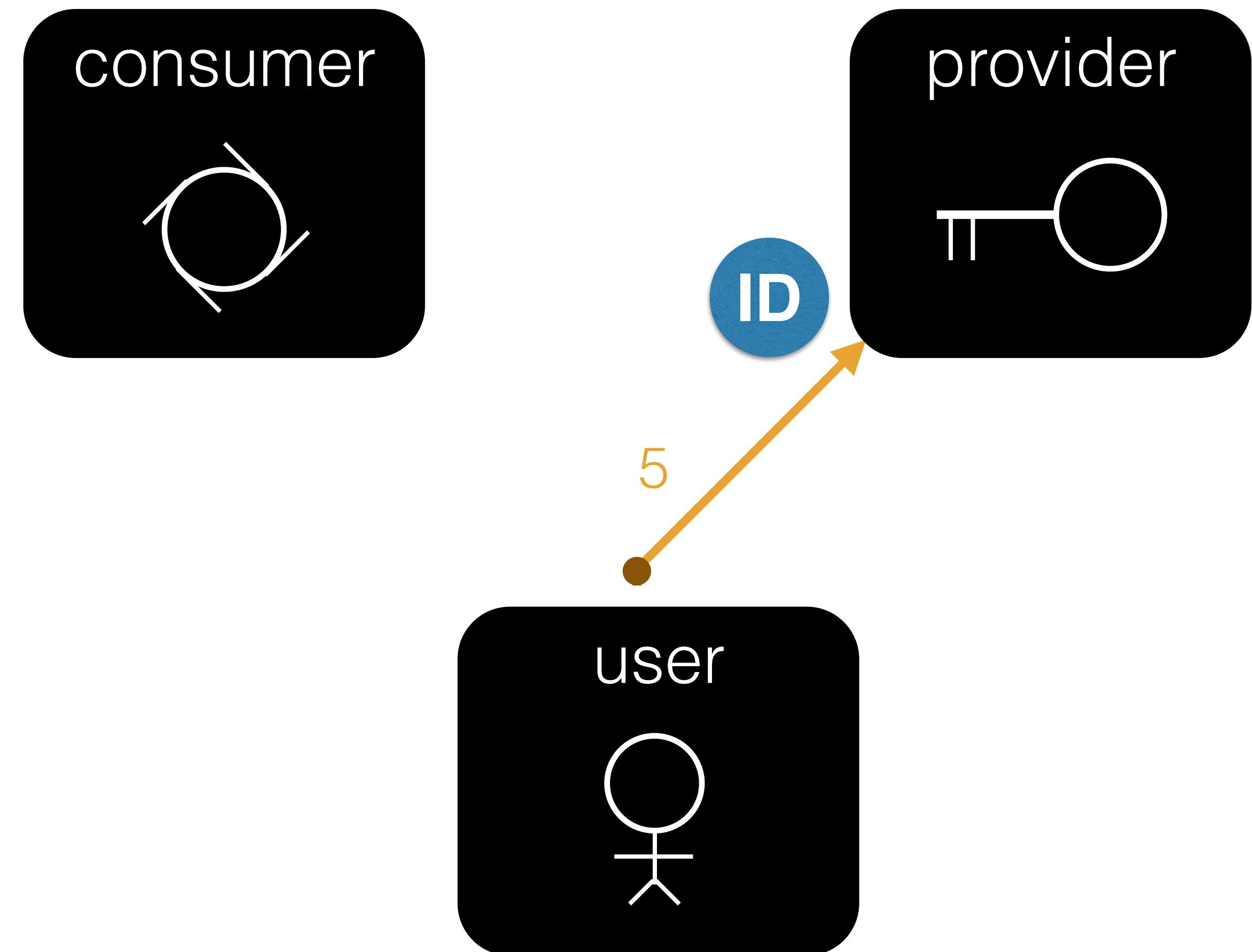


http://provider.com/oauth/authorize?client_id=123&callback_url=consumer.com/confirm&scope=read



OAUTH - USER AUTHENTICATION

5. Request: load login

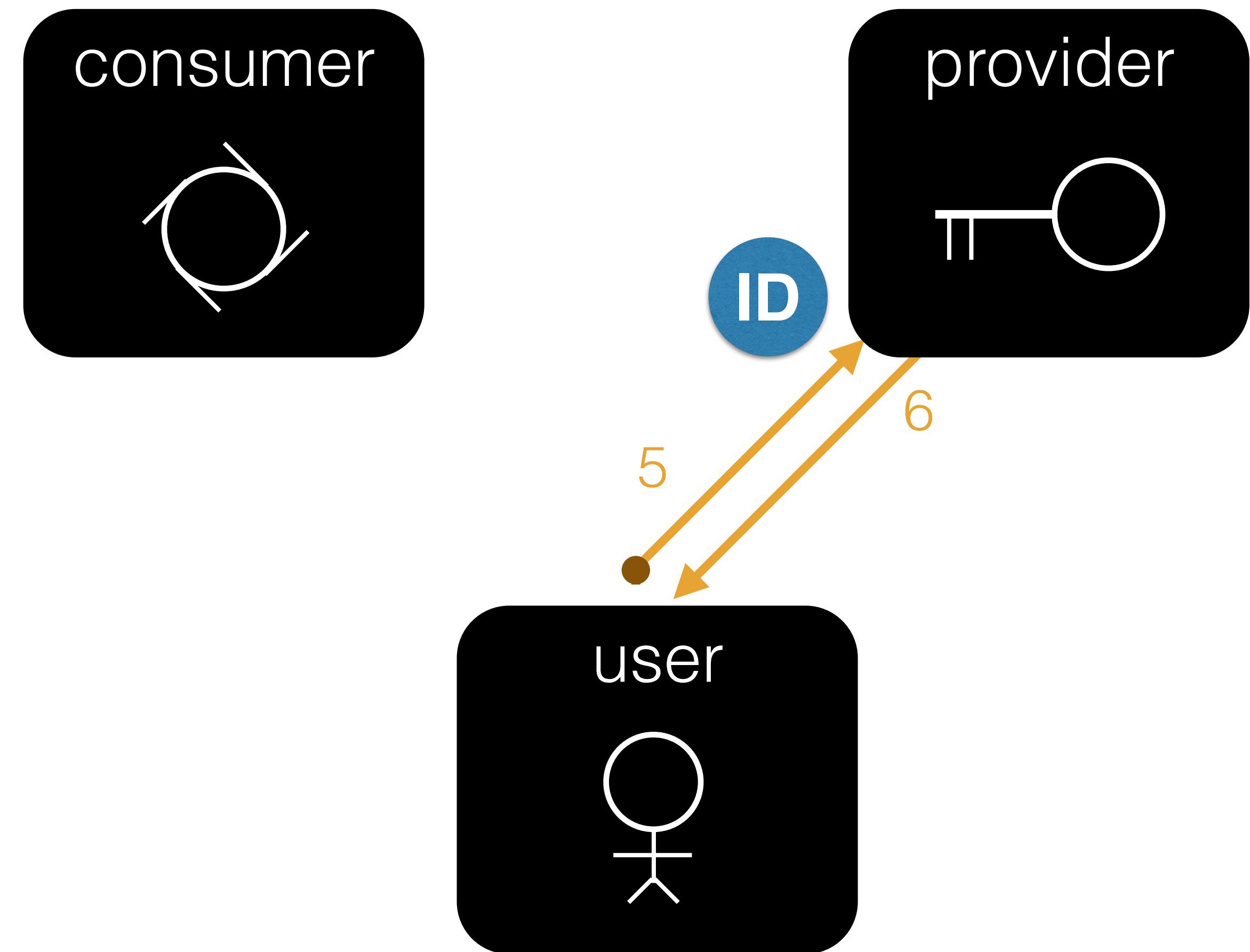


http://provider.com/oauth/authorize?client_id=123&callback_url=consumer.com/confirm&scope=read



OAUTH - USER AUTHENTICATION

5. Request: load login
6. Response: rendered login

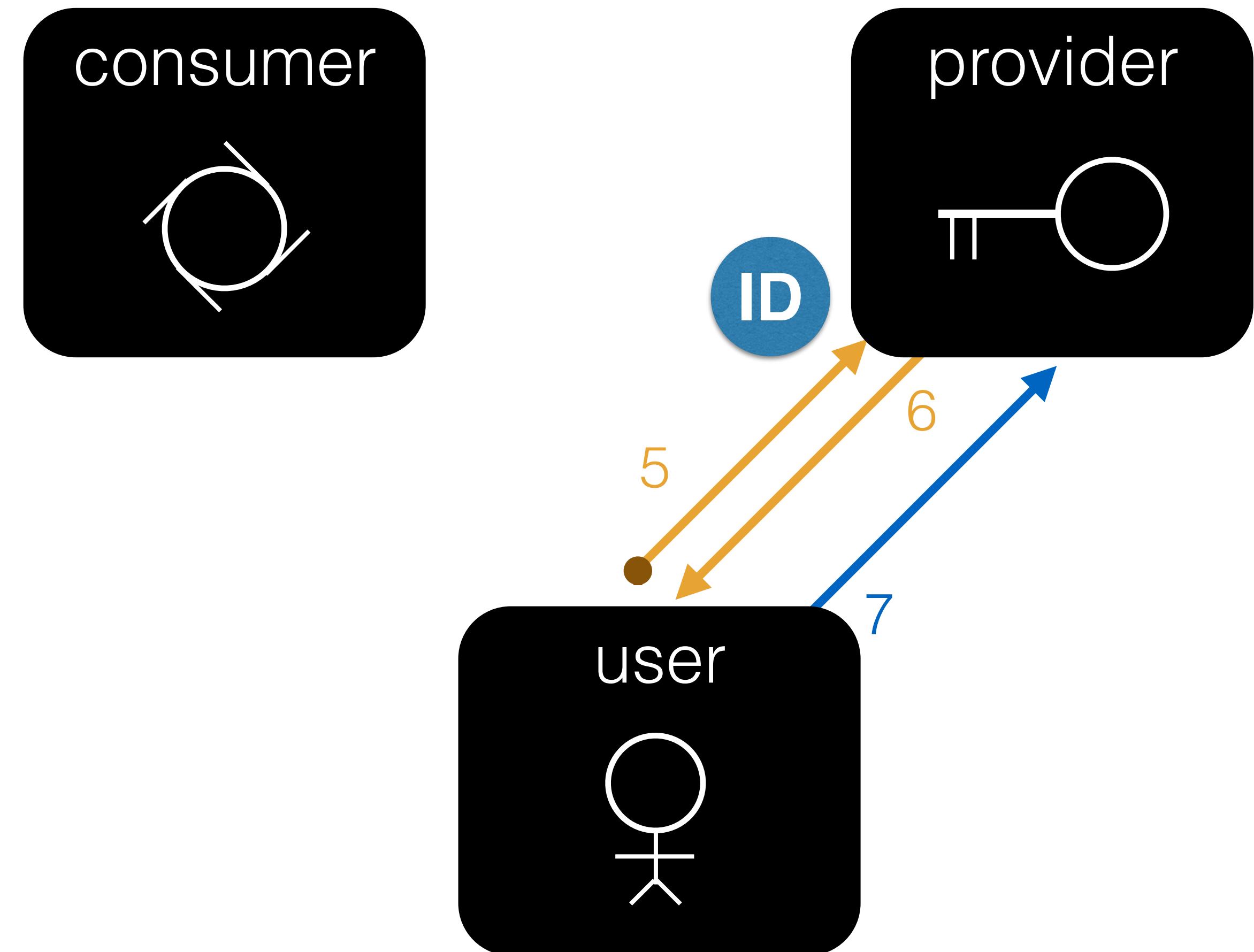


http://provider.com/oauth/authorize?client_id=123&callback_url=consumer.com/confirm&scope=read



OAUTH - USER AUTHENTICATION

- 5. Request: load login
- 6. Response: rendered login
- 7. Request: login to provider

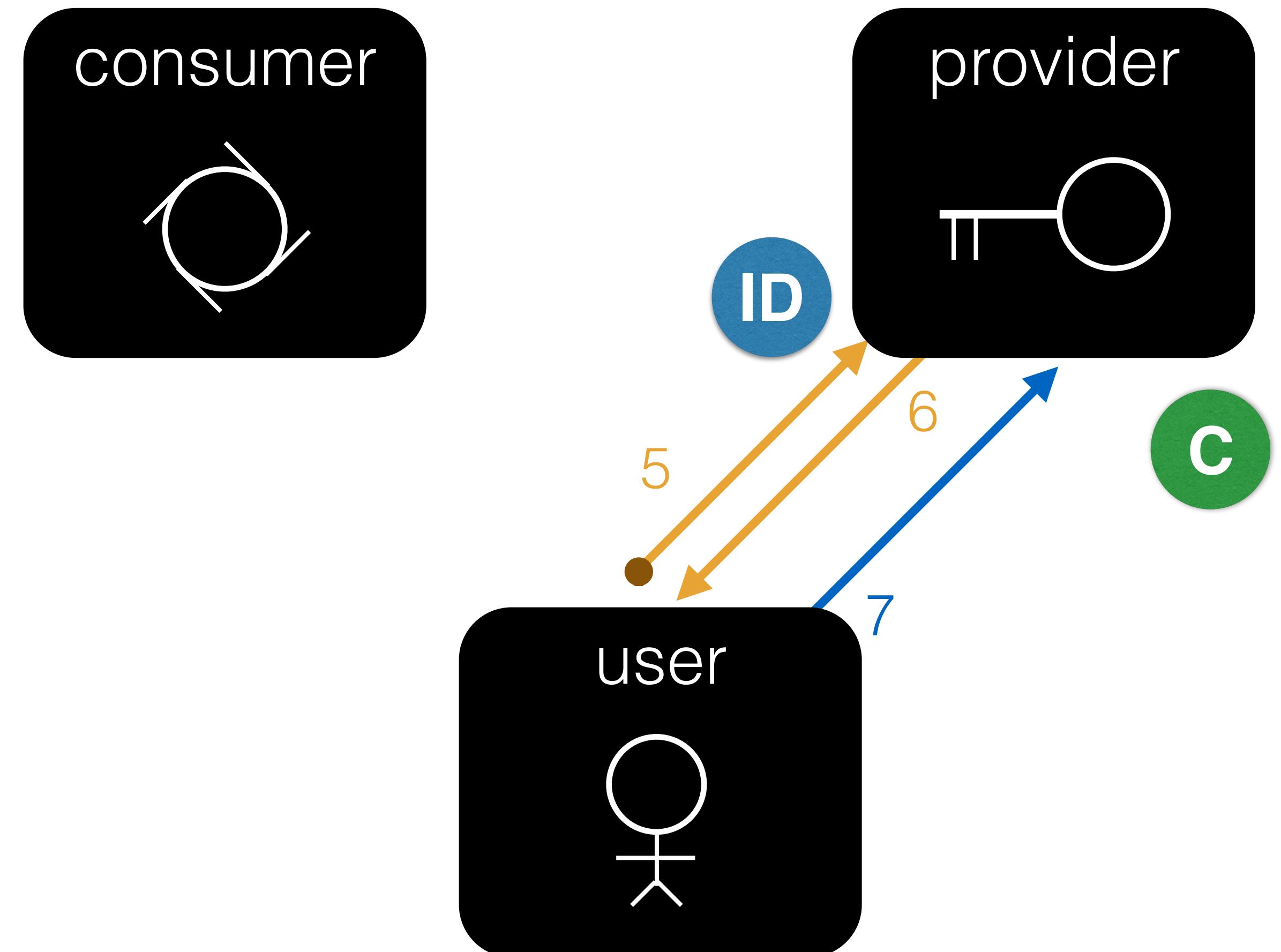


http://provider.com/oauth/authorize?client_id=123&callback_url=consumer.com/confirm&scope=read



OAUTH - USER AUTHENTICATION

- 5. Request: load login
- 6. Response: rendered login
- 7. Request: login to provider



<http://consumer.com/confirm?authcode=789>

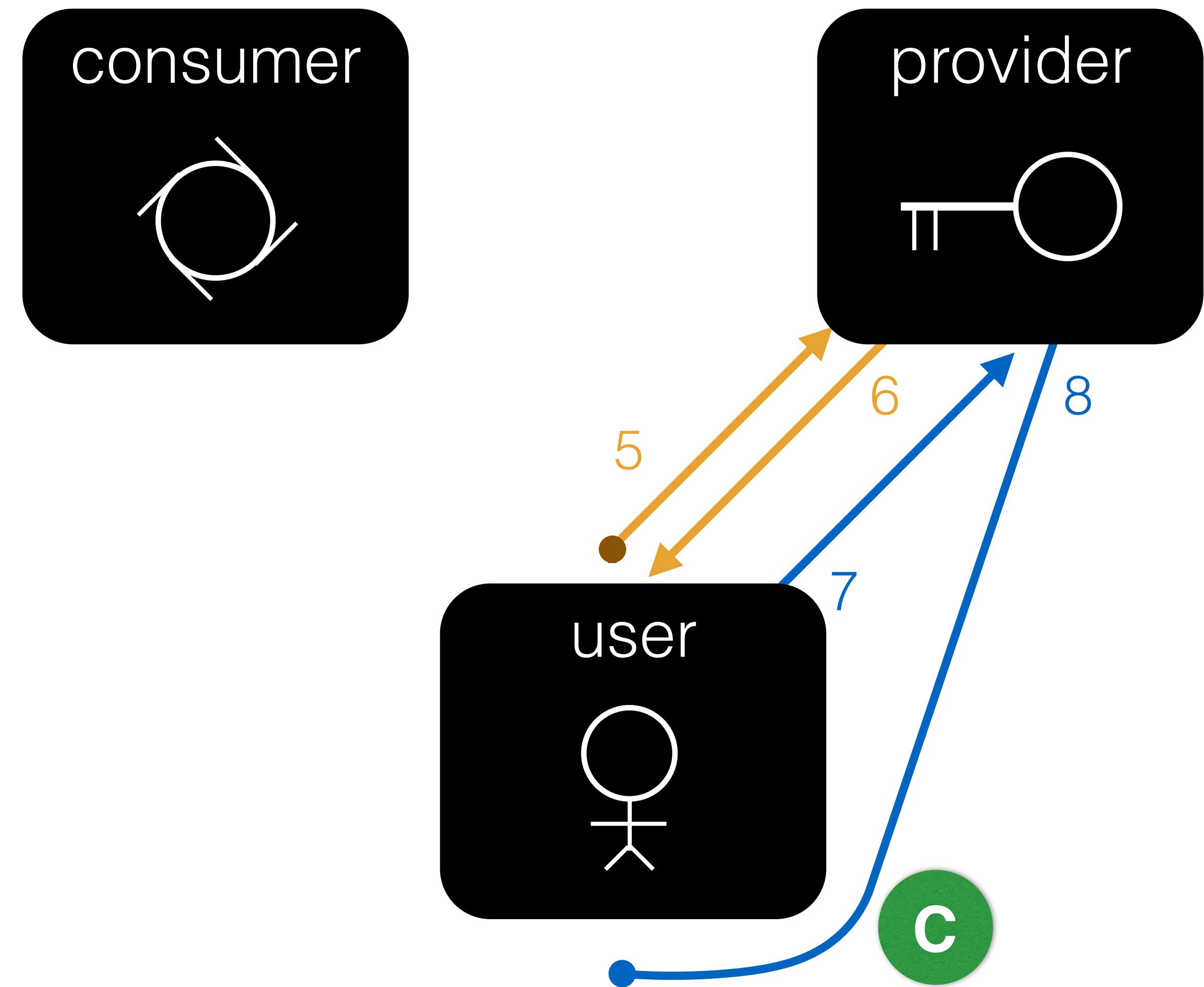
http://provider.com/oauth/authorize?client_id=123&callback_url=consumer.com/confirm&scope=read



OAUTH - USER AUTHENTICATION

- 5. Request: load login
- 6. Response: rendered login
- 7. Request: login to provider
- 8. Response: redirect callback URL

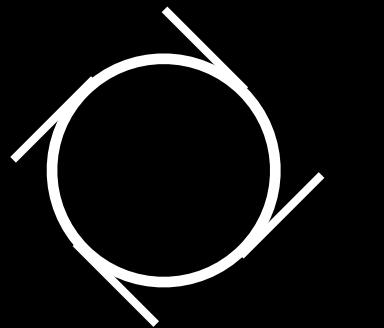
<http://consumer.com/confirm?authcode=789>



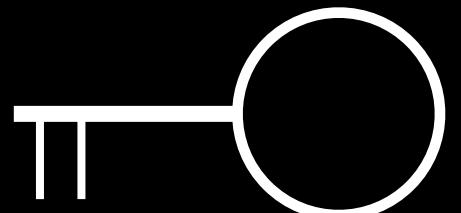


OAUTH - APP AUTHENTICATION

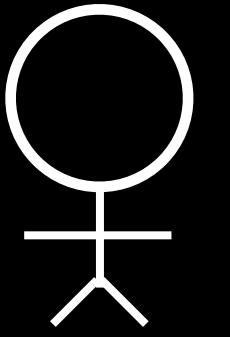
consumer



provider



user



<http://consumer.com/confirm?authcode=789>

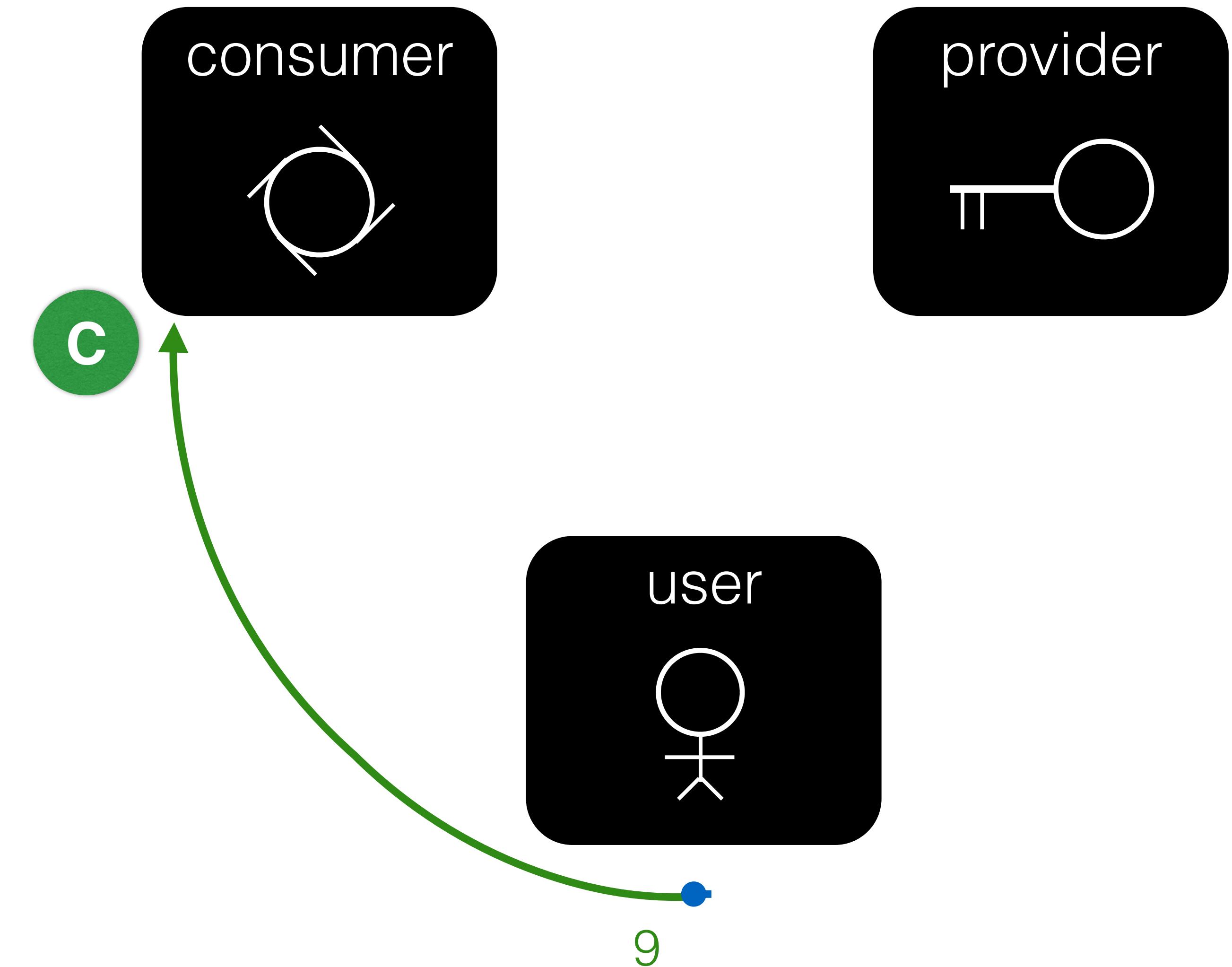




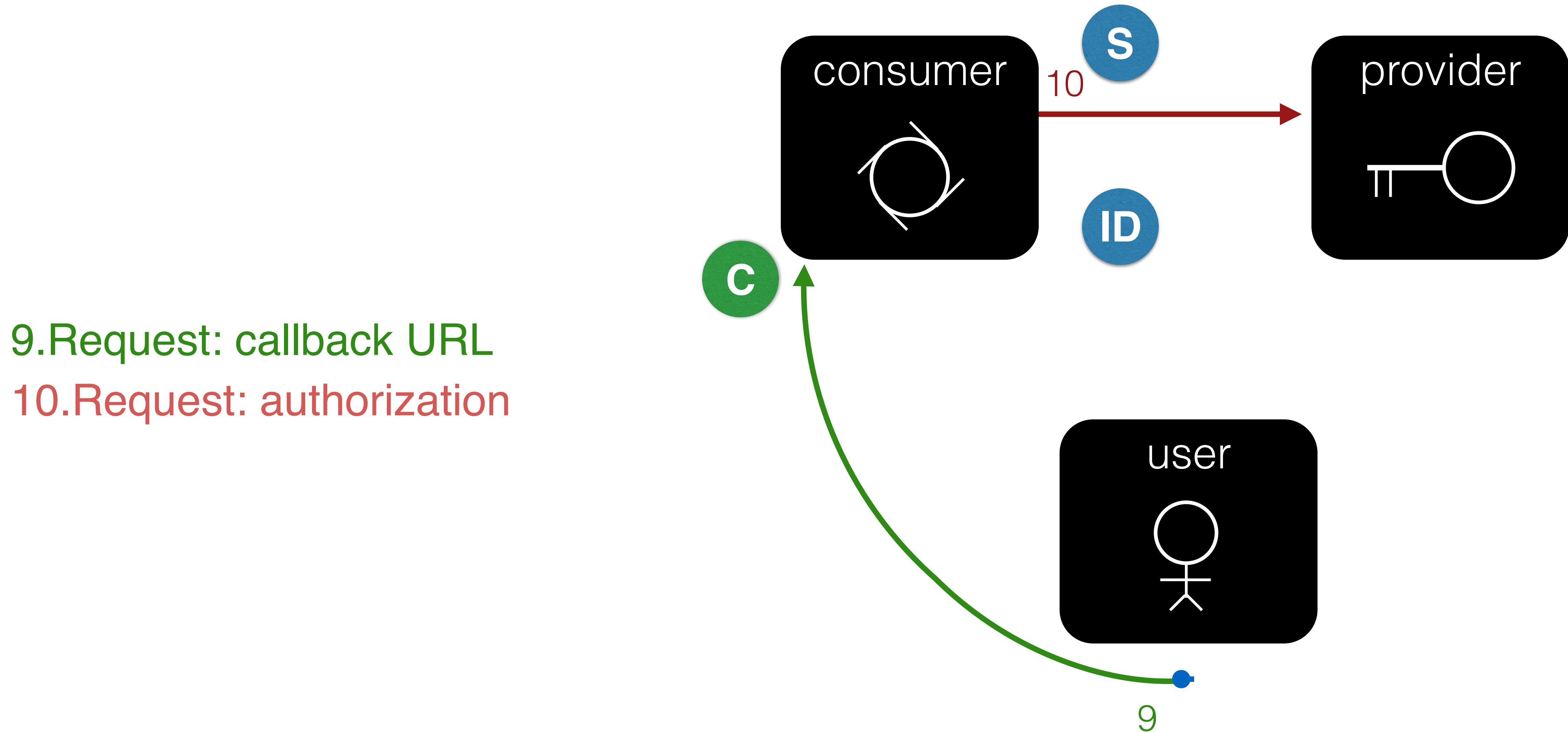
OAUTH - APP AUTHENTICATION

9. Request: callback URL

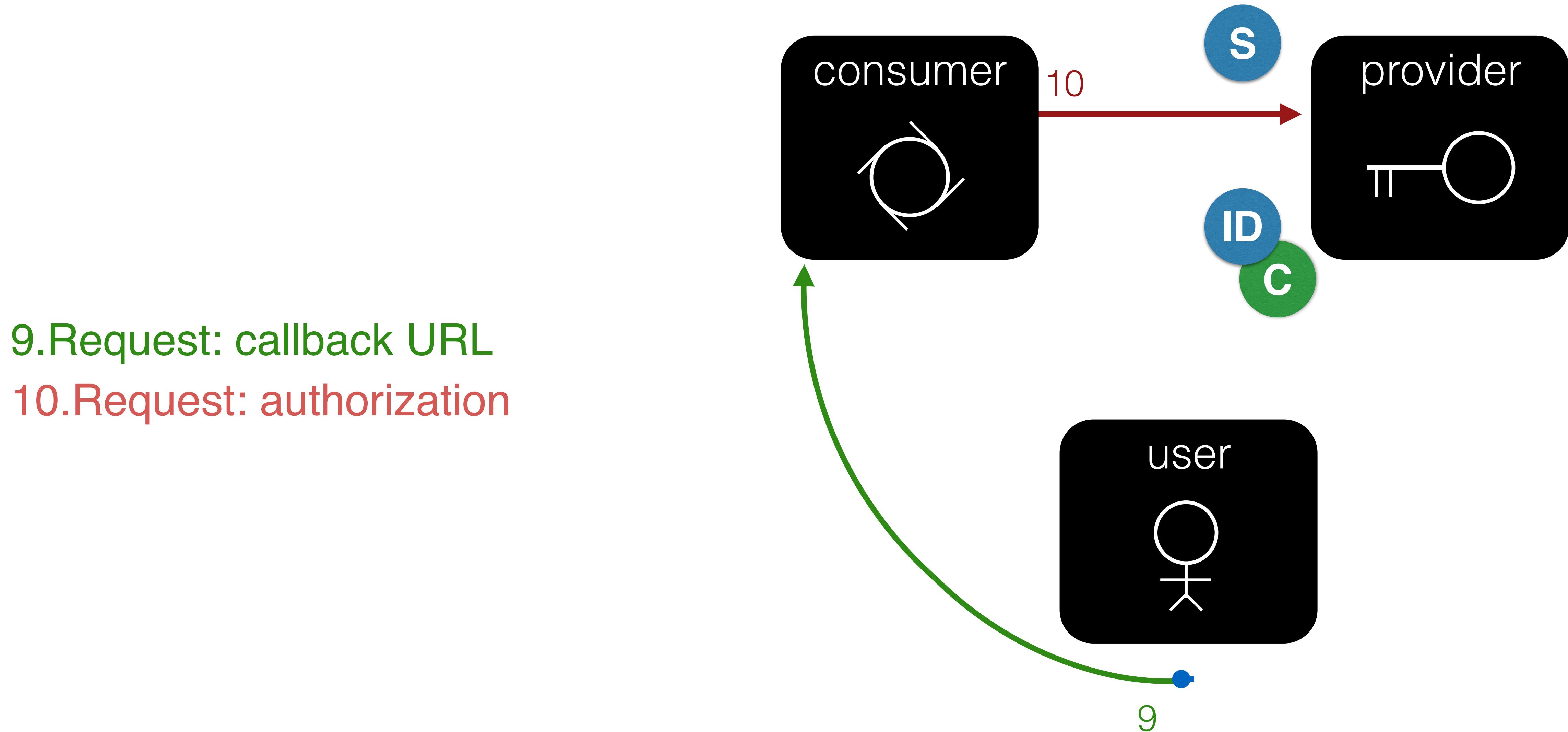
<http://consumer.com/confirm?authcode=789>



OAUTH - APP AUTHENTICATION



OAUTH - APP AUTHENTICATION



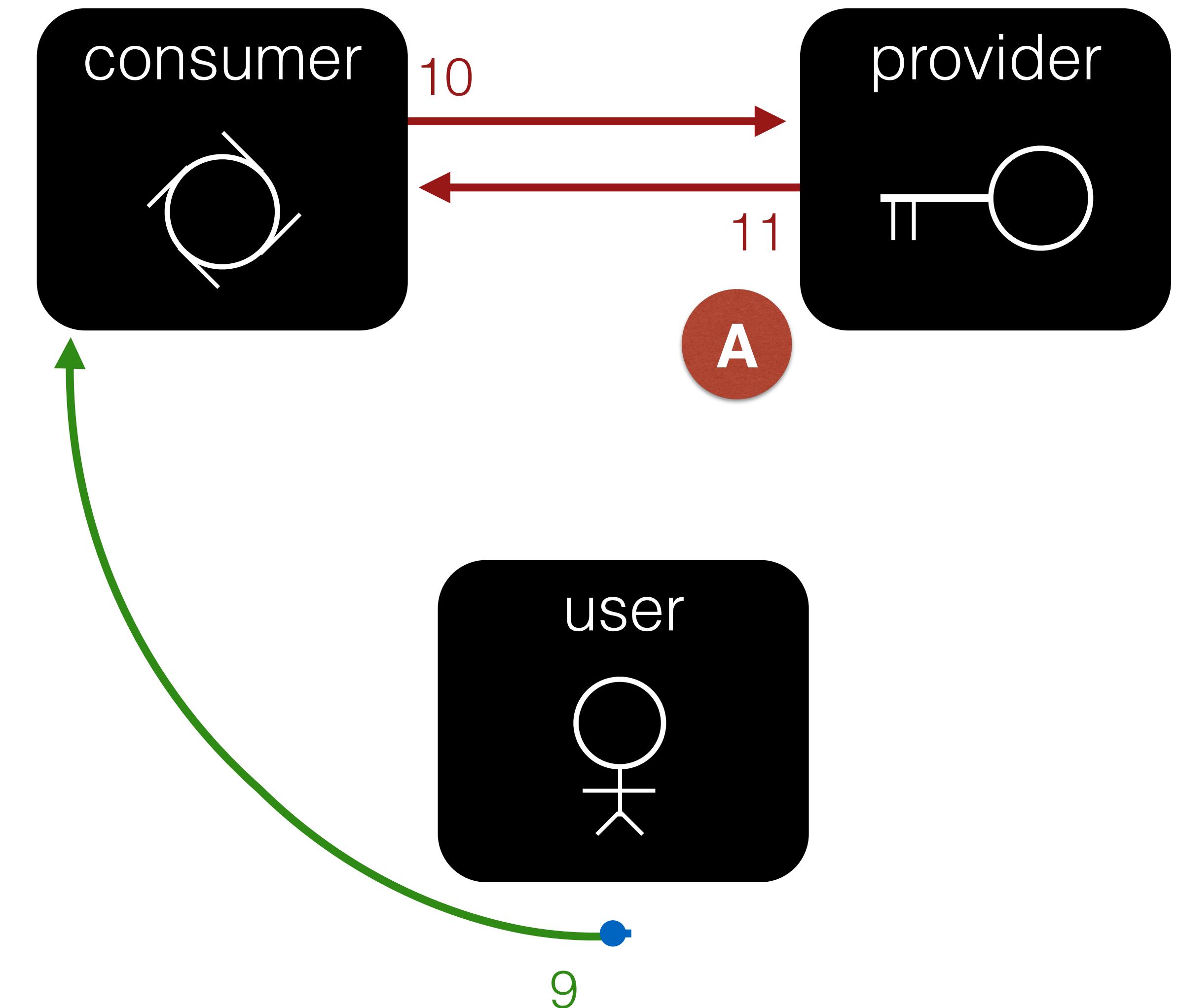
9.Request: callback URL

10.Request: authorization



OAUTH - APP AUTHENTICATION

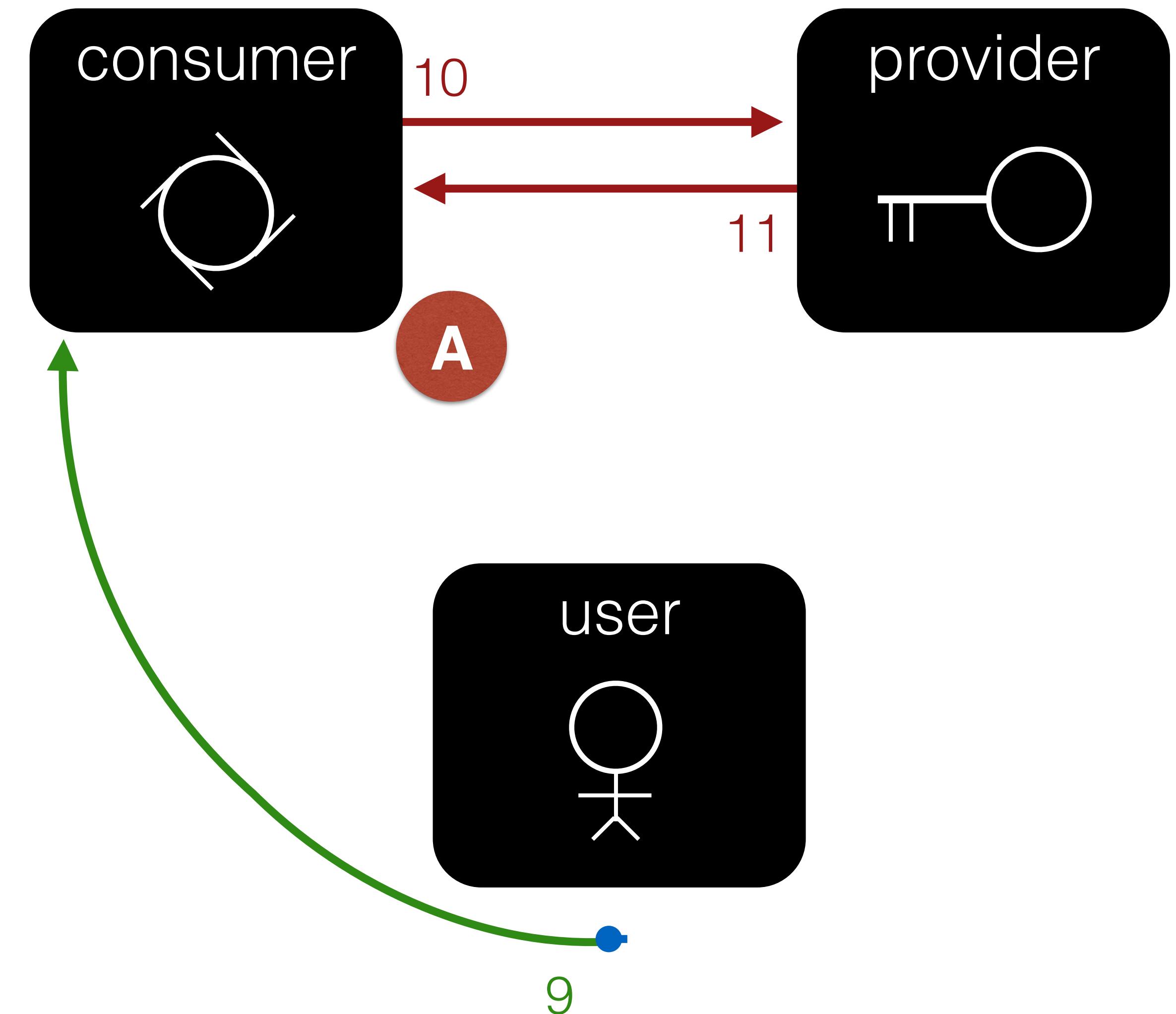
- 9. Request: callback URL
- 10. Request: authorization
- 11. Response: access token





OAUTH - APP AUTHENTICATION

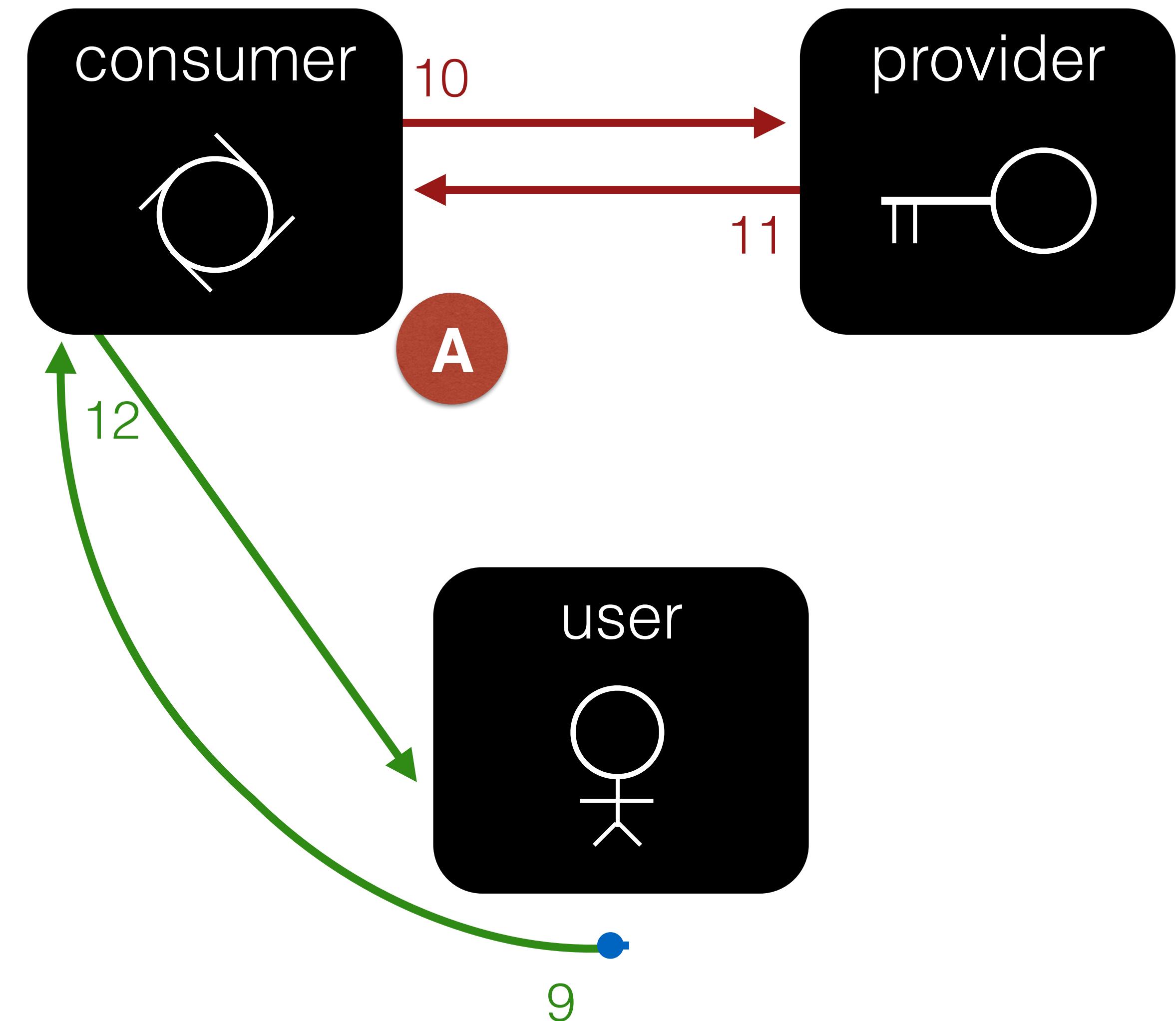
- 9. Request: callback URL
- 10. Request: authorization
- 11. Response: access token





OAUTH - APP AUTHENTICATION

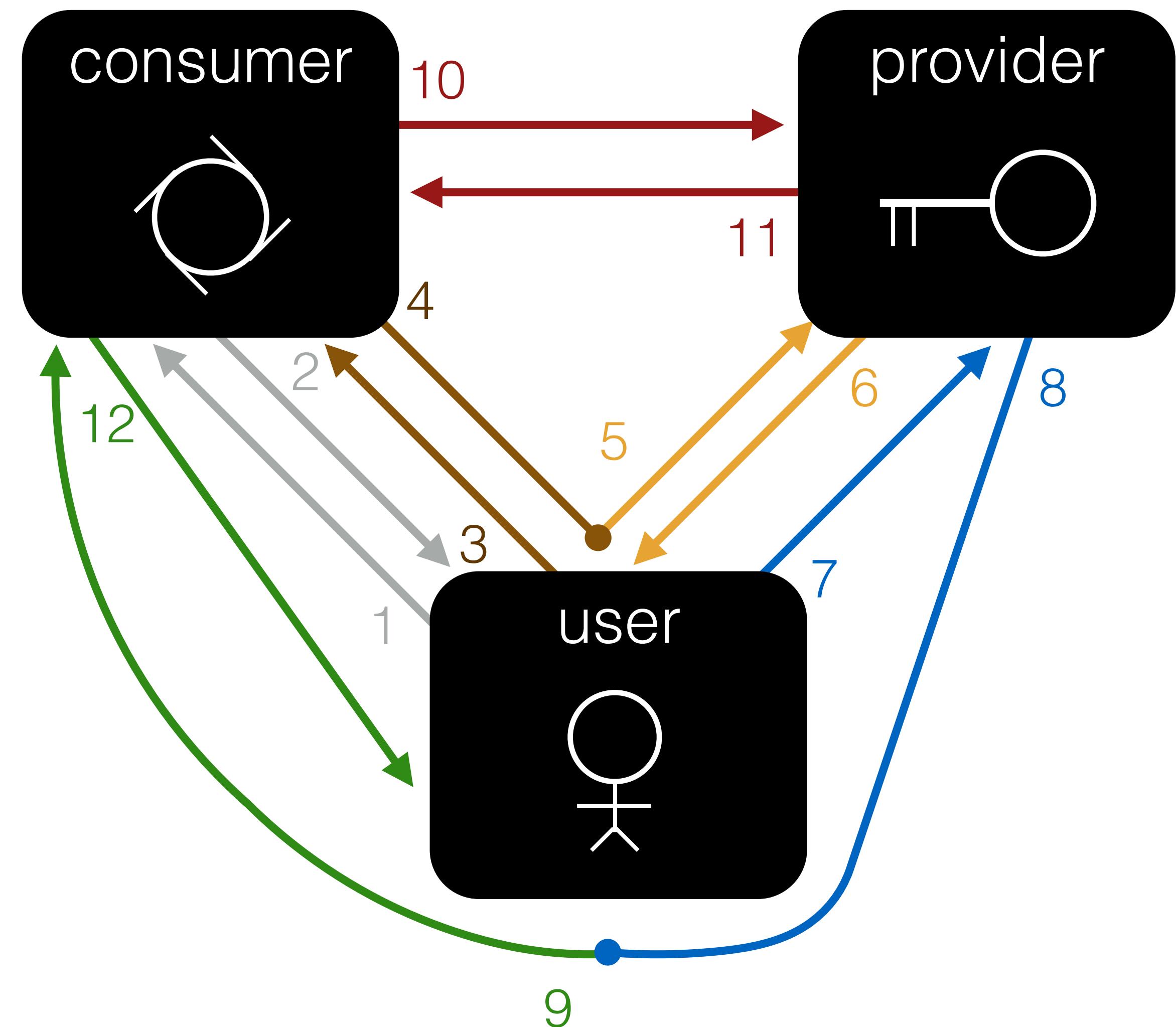
- 9. Request: callback URL
- 10. Request: authorization
- 11. Response: access token
- 12. Response: yay!





OAUTH - ALL OF IT

1. Request (user to app): load login page
2. Response (app to user): rendered login page
3. Request (user to app): allow app to use provider as me [user petitions app for a special contract to allow the app to do certain things on the user's behalf]
4. Response (app to user): redirect to provider login, passing along (to provider) an app id, a success "callback URL", and a permissions "contract" [app transfers this petition to provider]
5. Request (implicit, user to provider): load login page
6. Response (provider to user): rendered login page
7. Request (user to provider): login to provider [the user signs the contract]
8. Response (provider to user): on success, redirect to callback URL, passing along a new temporary code [the provider approves the user's signature]
9. Request (implicit, user to app): initiate callback
10. Request (app to provider): request for authorization given temporary code and app secret key [the app signs the contract]
11. Response (provider to app): on success, passes back an access token [the provider approves the app's signature and puts the contract into effect]
12. Response (app to user): we're good to go!

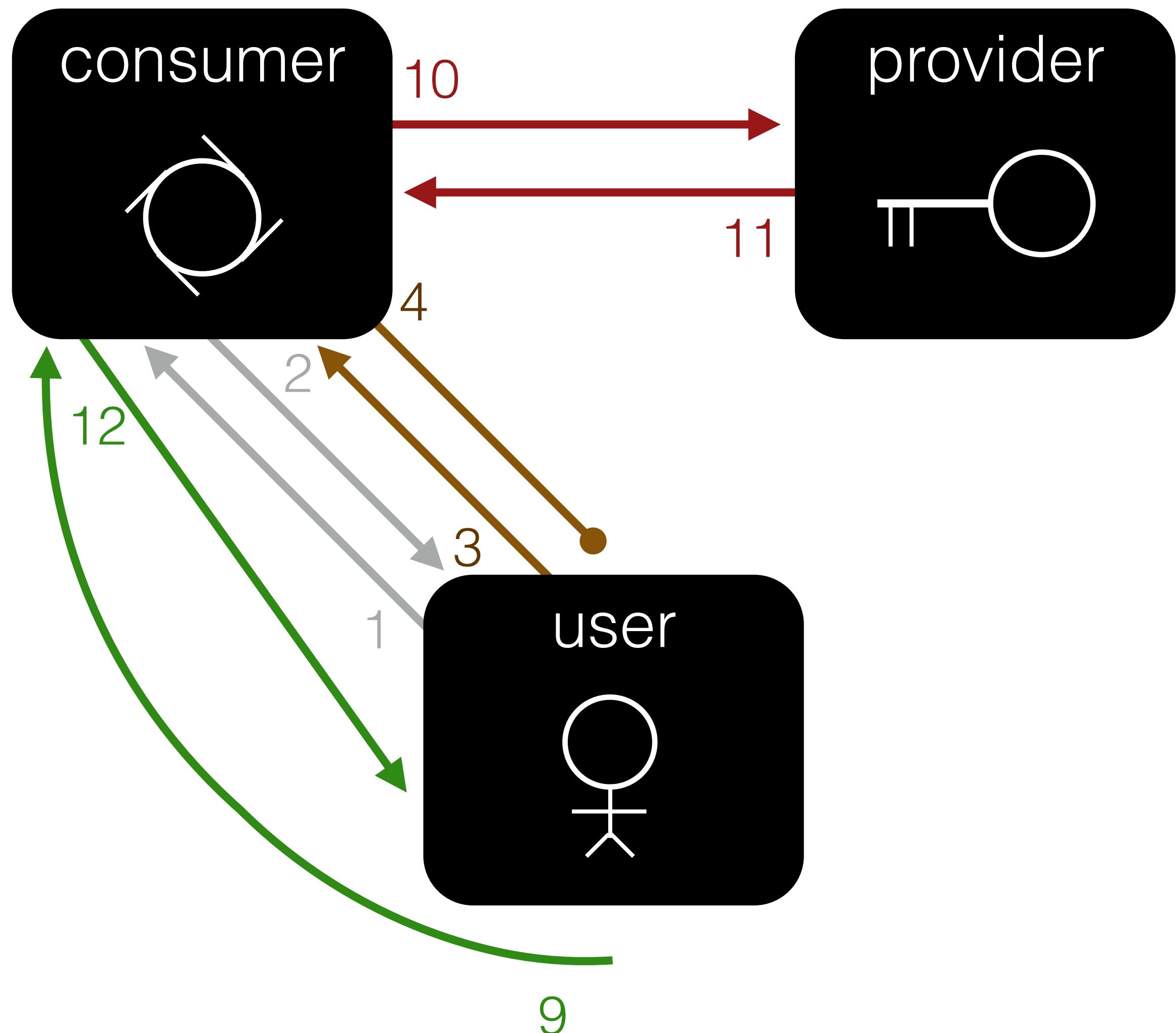




OAUTH - CONSUMER ROLE

1. Request (user to app): load login page
2. Response (app to user): rendered login page
3. Request (user to app): allow app to use provider as me [user petitions app for a special contract to allow the app to do certain things on the user's behalf]
4. Response (app to user): redirect to provider login, passing along (to provider) an app id, a success "callback URL", and a permissions "contract" [app transfers this petition to provider]

9. Request (implicit, user to app): initiate callback
10. Request (app to provider): request for authorization given temporary code and app secret key [the app signs the contract]
11. Response (provider to app): on success, passes back an access token [the provider approves the app's signature and puts the contract into effect]
12. Response (app to user): we're good to go!



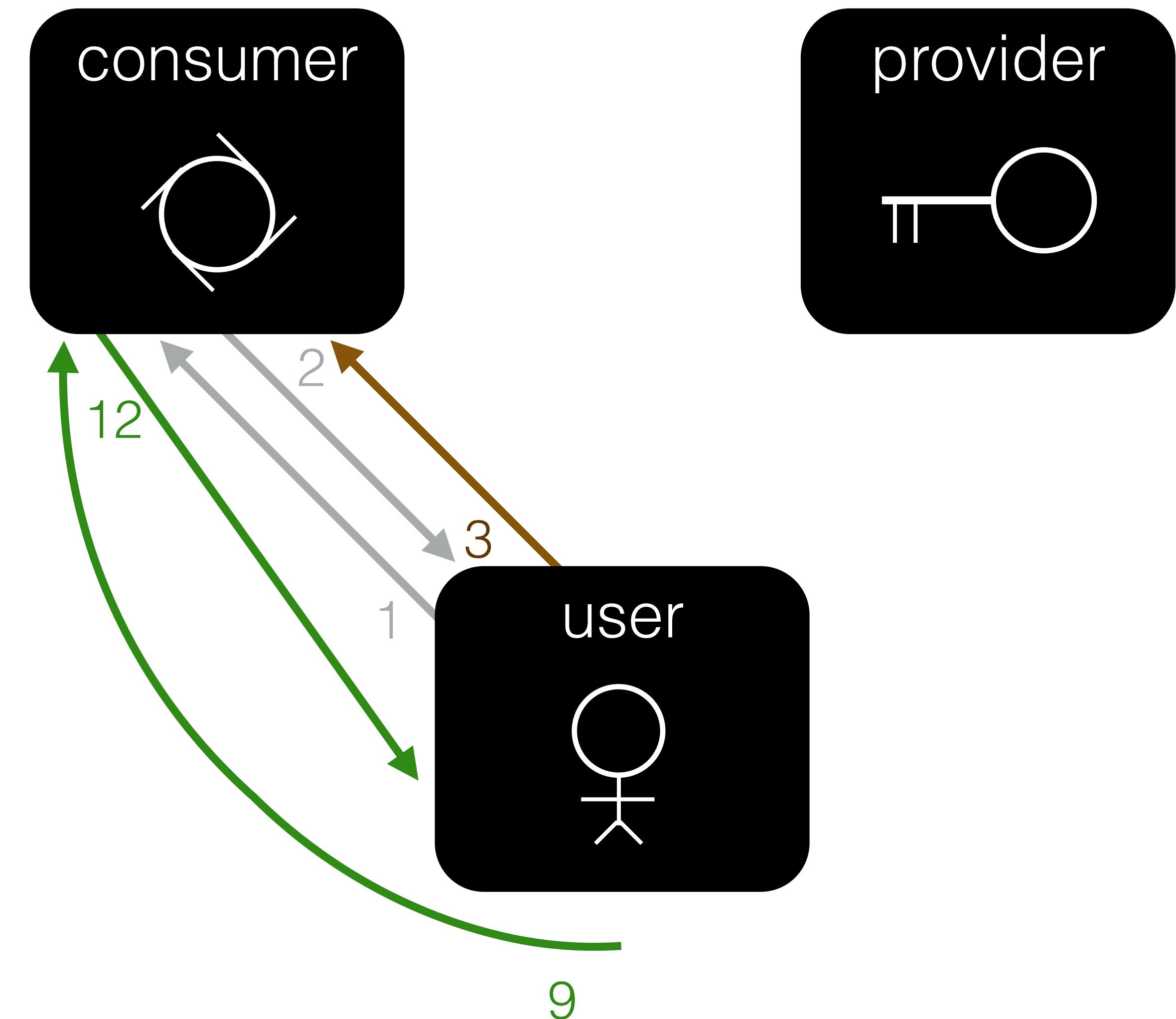


OAUTH - CONSUMER ROLE WITH PASSPORT

- 1.Request (user to app): load login page
- 2.Response (app to user): rendered login page
- 3.Request (user to app): allow app to use provider as me [user petitions app for a special contract to allow the app to do certain things on the user's behalf]

9.Request (implicit, user to app): initiate callback

12.Response (app to user): we're good to go!





AUTHER™