

Sessions

Web Development

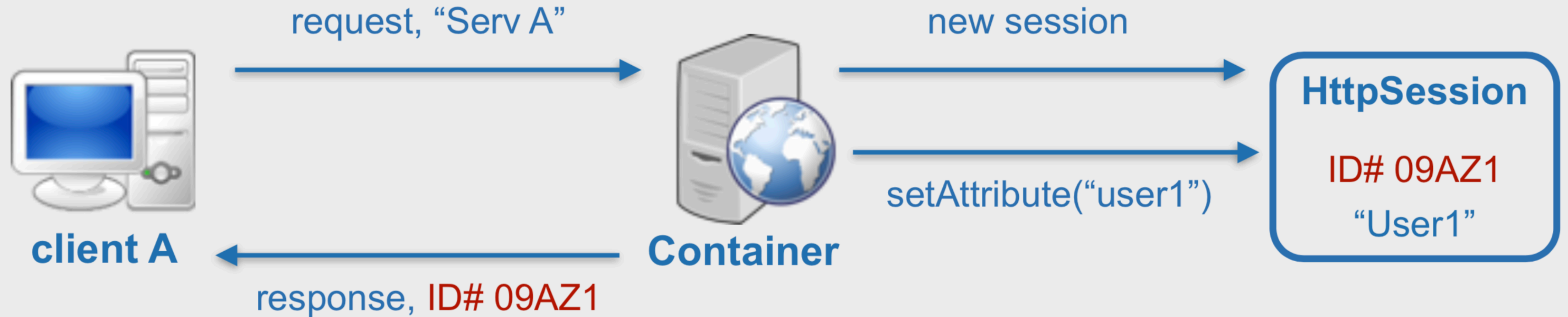
How to Make an Application out of a Web Page?

- On the internet, a web page is a web page is a web page...
 - If you surf from ./page1.html to ./page2.html these are two unique requests.
 - The server doesn't know anything about the fact that both pages are visited by the same user.
- Sessions are the technique used to logically group several requests into a “group” (called a session)
 - If you start a session, the server will know that it's still the same user who surfed from ./page1.html to ./page2.html

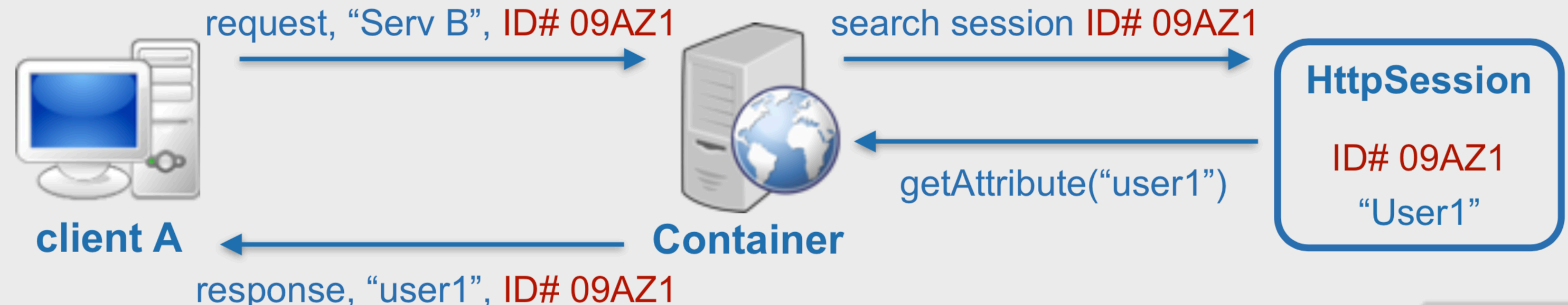


Session Tracking

First Request

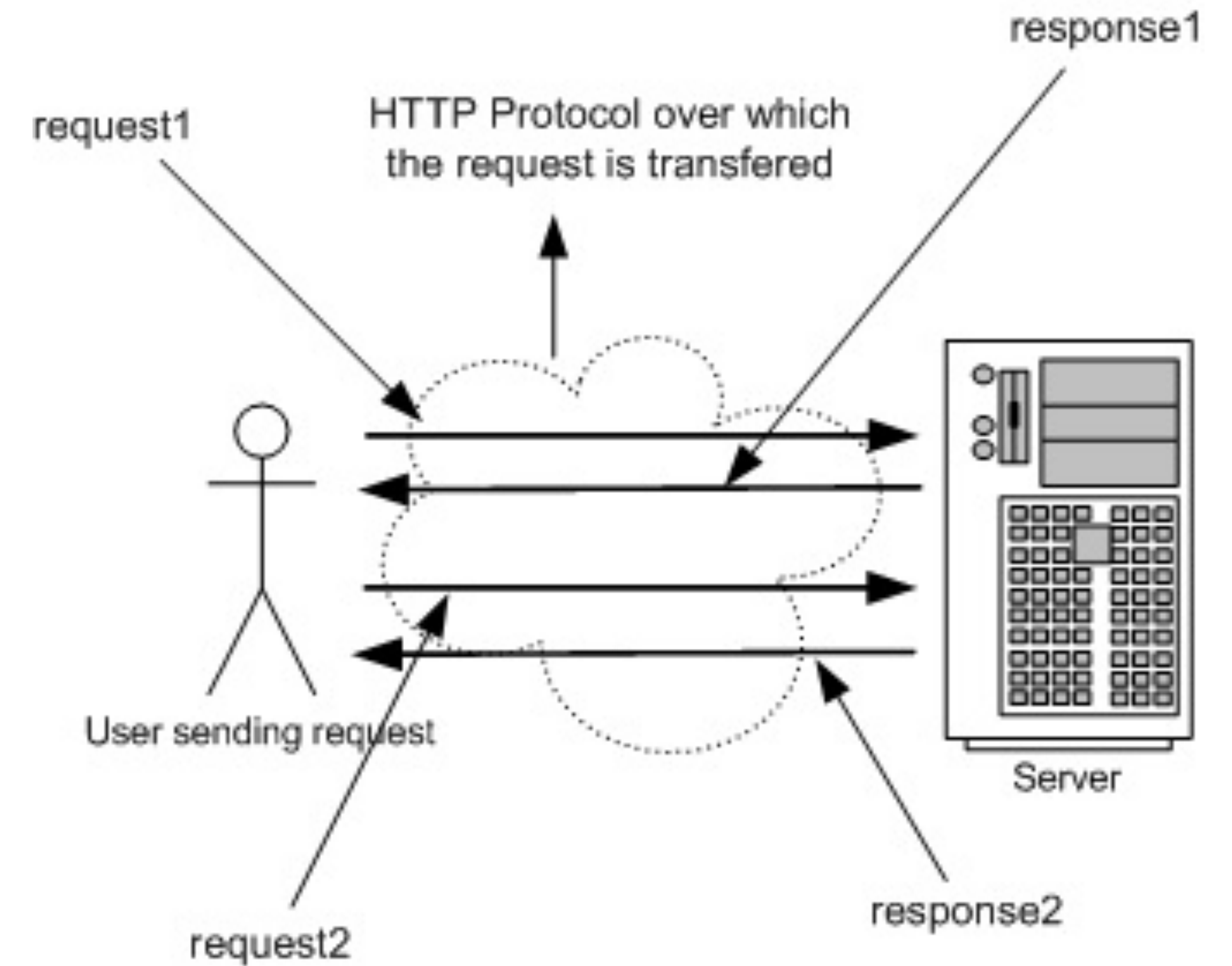


Subsequent Request



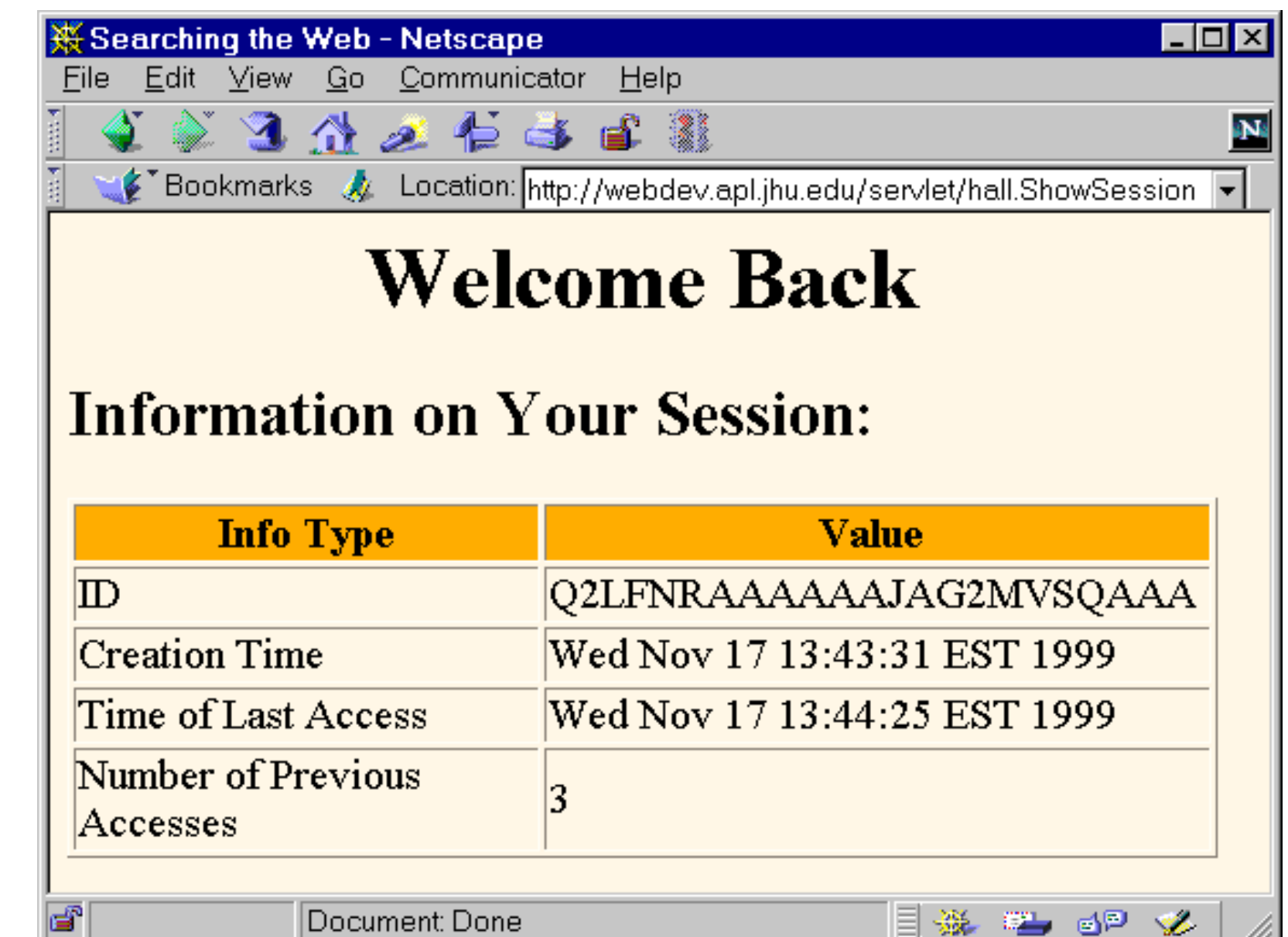
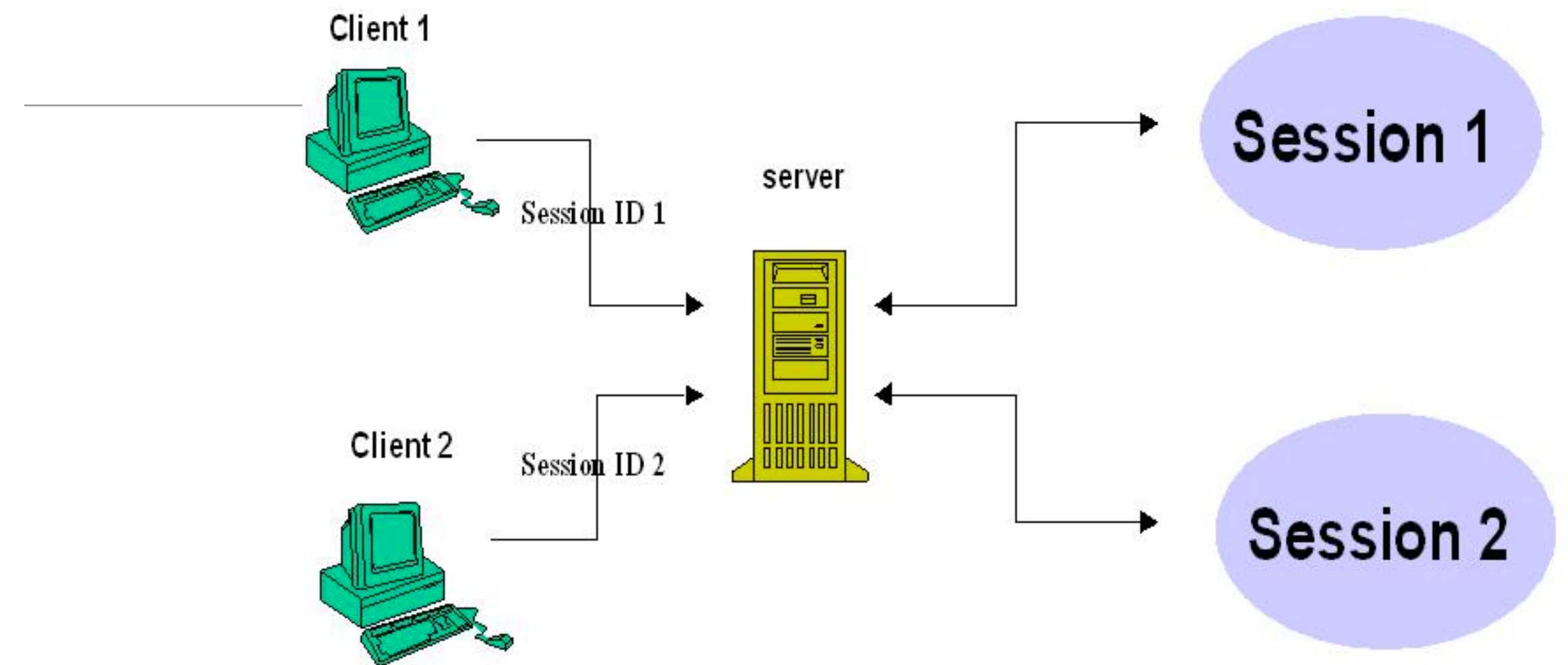
Sessions

- HTTP itself is “stateless”
 - no state stored on the server between requests from the same client
- but many web apps are stateful
 - necessary to connect requests from the same user / browser / browser-window, e.g. shopping cart, appointments calendar etc...
- *Session*
 - multiple requests performed in a stateful context
- *Session tracking*
 - technique that allows sessions in stateless environments



- User surfs to http://demo.com
- Server (on 1st request / if no sessionID stored on client)
 - generates unique session id, which is mapped to ...
 - ... a session-object
 - stored in memory (lost on shutdown), in a file or in database
 - can contain anything (list of articles, game state, counters, ...)
- Session id is added to the response
- from now on:
 - each subsequent request from the same user (browser) must contain the session id ...
 - ... which is used by the server to map to the session-object
- No data gets stored on the client, except SessionID

Session Tracking



Session Tracking Techniques

- Cookie
- Hidden Form Field
- URL Rewriting
- Json Web Token (JWT)

Cookies

First Response



client A

Http Response

HTTP/1.1 200 OK
Location: <http://www.abcd.com/login>
Set-Cookie: JSESSIONID=09AZ1
Domain=.abcd.com;path=/;HttpOnly
.....



Container

Subsequent Requests



client A



Http Request

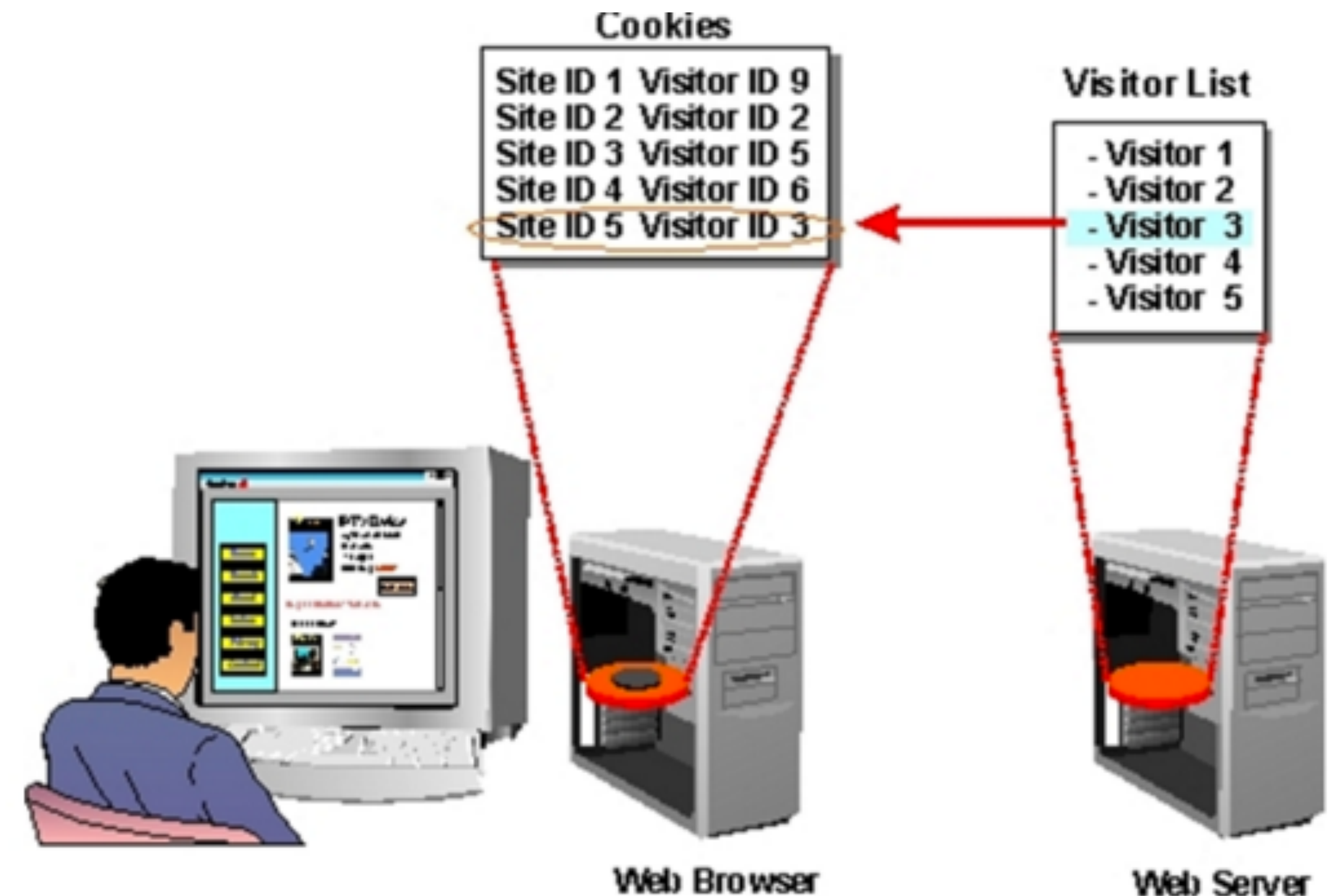
POST/login.do HTTP/1.1
Host: www.abcd.com
Cookie: JSESSIONID=09AZ1
.....



Container

Cookies

1. Server creates a cookie with session-id on first request
2. Server maps id to a new user-specific session object
3. The session-id is sent to the client with the first response
4. ..and automatically added by the browser on each further request (to the same address/domain/...)
5. Server receives request + cookie with session-id
6. Server maps session-id to session-object



donation-web cookie (in browser)

Inspector

Console

Debugger

Style Editor

Performance

Memory

Network

Storage

Accessibility

Cache Storage

Cookies

Indexed DB

Local Storage

Session Storage

+ ↺

Filter items

⌵

Name	Domain	Path	Expires on	Last accessed on	Value
donation-web	localhost	/	Thu, 31 Jan 2019 07:01:14 GMT	Wed, 30 Jan 2019 07:01:14 GMT	Fe26.2**78d4c93c186076e4cd271c2b1d33e5a1548cf79134789aaee69cd0a4389b7174

Filter values

▼ Data

donation-web: "Fe26.2**78d4c93c186076e4cd...LNHdXmQBk2v4cTgKTZKEkvBWU"

CreationTime: "Wed, 30 Jan 2019 07:01:14 GMT"

Domain: "localhost"

Expires: "Thu, 31 Jan 2019 07:01:14 GMT"

HostOnly: true

HttpOnly: true

LastAccessed: "Wed, 30 Jan 2019 07:01:14 GMT"

Path: "/"

Secure: false

sameSite: "Strict"

▼ Parsed Value

▼ donation-web: Array

0: "Fe26.2"

1: ""

2: "78d4c93c186076e4cd271c2b1d33e5a1548cf79134789aaee69cd0a4389b7174"

3: "EV_4-Ds7qYD3Qqz3YxyPIQ"

4: "TtPdRqA3xzll5DlVBjXG3nr9gMPiXx_2ThreofArK-7HMKL18neAgUCrmibC99K"

5: ""

6: "28a6a80b7dece305e6cfa1c7661d50cbfe1220f121e1c5687c7a72289c405ebc"

7: "zTN-pcJx0oYtzS4oQhLNHdXmQBk2v4cTgKTZKEkvBWU"

length: 8

__proto__: Array

9

hapi-auth-cookie

- ‘Official’ cookie plugin for Hapi

hapi-auth-cookie

hapi Cookie authentication plugin

build failing

Lead Maintainer: [Eran Hammer](#)

Cookie authentication provides simple cookie-based session management. The user has to be authenticated via other means, typically a web form, and upon successful authentication the browser receives a reply with a session cookie. The cookie uses [Iron](#) to encrypt and sign the session content.

Subsequent requests containing the session cookie are authenticated and validated via the provided `validateFunc` in case the cookie's encrypted content requires validation on each request.

It is important to remember a couple of things:

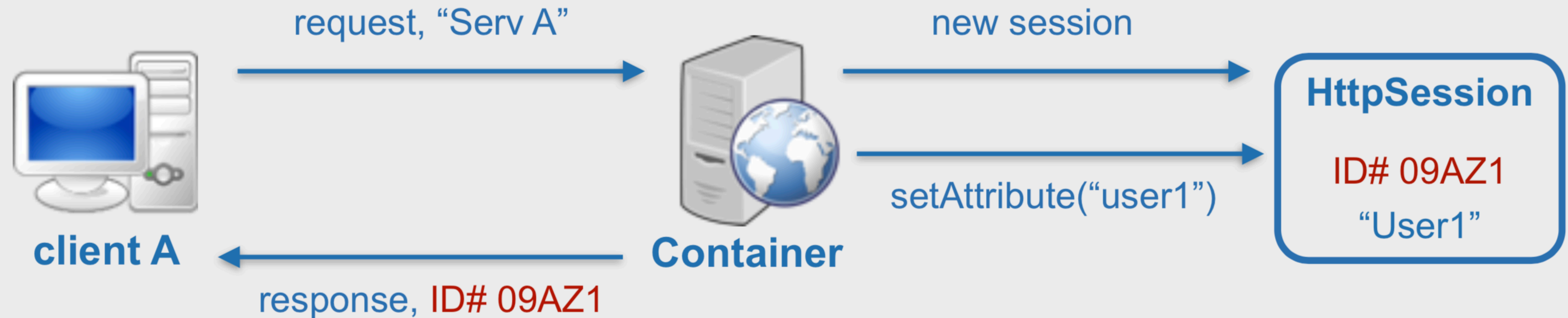
1. Each cookie operates as a bearer token and anyone in possession of the cookie content can use it to impersonate its true owner.
2. Cookies have a practical maximum length. All of the data you store in a cookie is sent to the browser. If your cookie is too long, browsers may not set it. Read more [here](#) and [here](#). If you need to store more data, store a small amount of identifying data in the cookie and use that as a key to a server-side cache system.

The `'cookie'` scheme takes the following options:

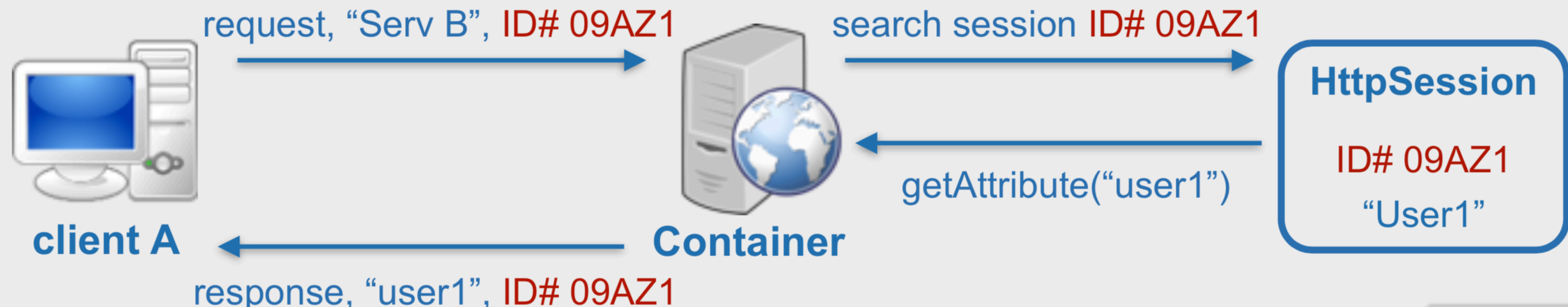
- `cookie` - the cookie name. Defaults to `'sid'`.
- `password` - used for Iron cookie encoding. Should be at least 32 characters long.
- `ttl` - sets the cookie expires time in milliseconds. Defaults to single browser session (ends when browser closes). Required when `keepAlive` is `true`.
- `domain` - sets the cookie Domain value. Defaults to none.
- `path` - sets the cookie path value. Defaults to `/`.
- `clearInvalid` - if `true`, any authentication cookie that fails validation will be marked as expired in the response and cleared. Defaults to `false`.
- `keepAlive` - if `true`, automatically sets the session cookie after validation to extend the current session for a new `ttl` duration. Defaults to `false`.

URL Rewrite

First Request



Subsequent Request



URL Rewrite

- Server adds the session-id to all links the user can follow
 - `http://server/myhome`
- is changed to
 - `http://server/myhome?sessionid=123`
- session-id must be dynamically added
 - functionality usually offered by scripting frameworks

Hidden Form Fields

- In HTML, we can define "hidden" fields in a form
 - `<input type="hidden" name="sessionid" value="123">`
- These fields are not visible and cannot be changed by the client
- Usage:
 - server creates a session-object for each client and generates a unique ID
 - When HTML documents are created and sent back, the hidden form field is automatically generated containing the actual ID
 - Upon form submit, the session ID is automatically sent back to the server
 - The server can associate this call with an already existing session

Hidden Form Filed Example

Donation

Donate


Report

Settings

Logout

Select Amount ▾
☐ Paypal
☐ Direct

Donate




```
<form action="/donate" method="POST">
  <input type="hidden" name="userID" value="2354515">
  <div class="ui dropdown" name="amount">
    <input type="hidden" name="amount">
    <div class="text">Select Amount</div>
    <i class="ui dropdown icon"></i>
    <div class="menu">
      <div class="item">50</div>
      <div class="item">100</div>
      <div class="item">1000</div>
    </div>
  </div>
  <div class="grouped inline fields">
    <div class="field">
      <div class="ui radio checkbox">
        <input type="radio" name="method" value="paypal">
        <label>Paypal</label>
      </div>
    </div>
    <div class="field">
      <div class="ui radio checkbox">
        <input type="radio" name="method" value="direct">
        <label>Direct</label>
      </div>
    </div>
  </div>
  <button class="ui blue submit button">Donate</button>
</form>
```

Hidden Form Filed Example

Donation Donate Report Settings Logout

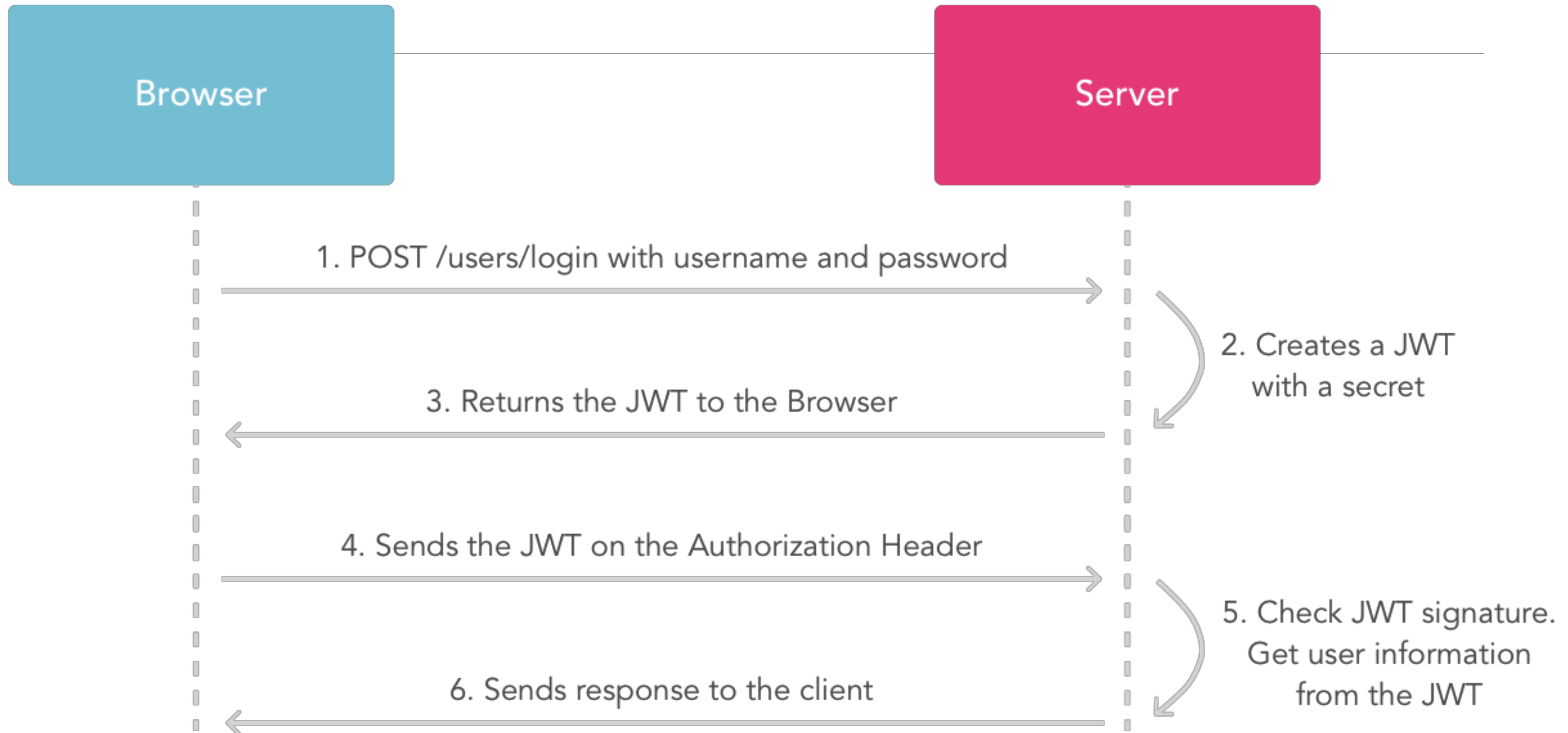
Select Amount ▾
☐ Paypal
☐ Direct
Donate



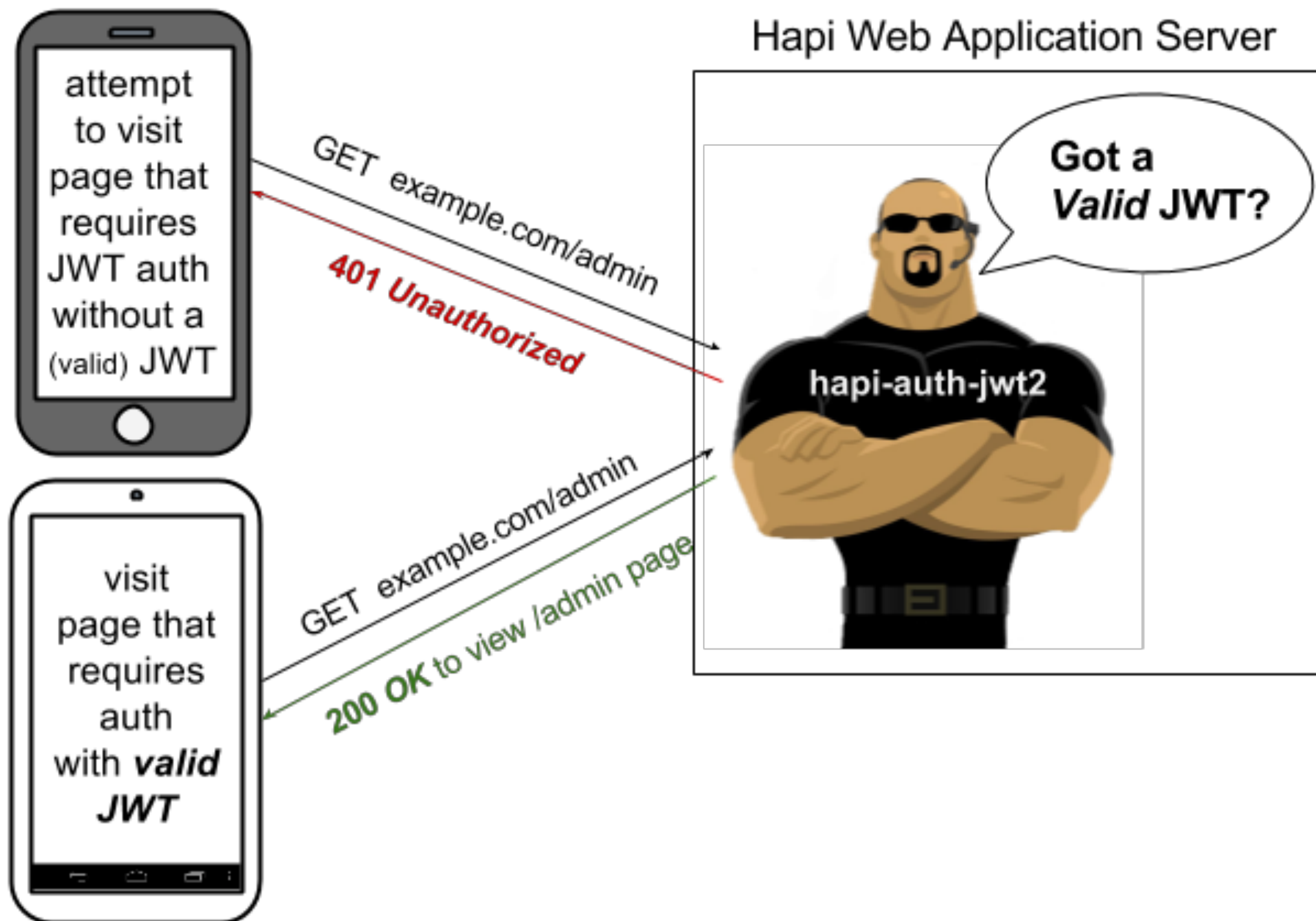
```
<form action="/donate" method="POST">
  <input type="hidden" name="userID" value="2354515">
  <div class="ui dropdown" name="amount">
    <input type="hidden" name="amount">
    <div class="text">Select Amount</div>
    <i class="ui dropdown icon"></i>
    <div class="menu">
      <div class="item">50</div>
      <div class="item">100</div>
      <div class="item">1000</div>
    </div>
  </div>
  <div class="grouped inline fields">
    <div class="field">
      <div class="ui radio checkbox">
        <input type="radio" name="method" value="paypal">
        <label>Paypal</label>
      </div>
    </div>
  </div>
</form>
```

Json Web Token

- An open standard that defines a compact and self-contained way for securely transmitting information between parties as a JSON object.
- **Compact:** Because of its smaller size, JWTs can be sent through an URL, POST parameter, or inside an HTTP header.
- **Self-contained:** The payload contains all the required information about the user, avoiding the need to query the database more than once.
- **Authentication:** Once the user is logged in, each subsequent request will include the JWT, allowing the user to access routes, services, and resources that are permitted with that token.
- **Information Exchange:** JSON Web Tokens are a good way of securely transmitting information between parties, because they can be signed.



hapi-auth-jwt2



This node.js module (Hapi plugin) lets you use JSON Web Tokens (JWTs) for authentication in your [Hapi.js](#) web application.

If you are totally new to JWTs, we wrote an introductory post explaining the concepts & benefits:

<https://github.com/dwyl/learn-json-web-tokens>

If you (or anyone on your team) are unfamiliar with **Hapi.js** we have a quick guide for that too:

<https://github.com/dwyl/learn-hapi>

Usage

We tried to make this plugin as user (developer) friendly as possible, but if anything is unclear, please submit any questions as issues on GitHub: <https://github.com/dwyl/hapi-auth-jwt2/issues>

Install from NPM

```
npm install hapi-auth-jwt2 --save
```

Example

This basic usage example should help you get started:

```
const Hapi = require('hapi');

const people = { // our "users database"
  1: {
    id: 1,
    name: 'Jen Jones'
  }
};

// bring your own validation function
const validate = async function (decoded, request) {

  // do your checks to see if the person is valid
  if (!people[decoded.id]) {
    return { isValid: false };
  }
  else {
    return { isValid: true };
  }
};
```

Web Frameworks

- Cookies generally preferred.
- However, framework may try to ‘abstract away’ specific session management technology, and deliver simpler abstraction to the programmer
- Framework may in fact be able to switch between different techniques depending on circumstances.