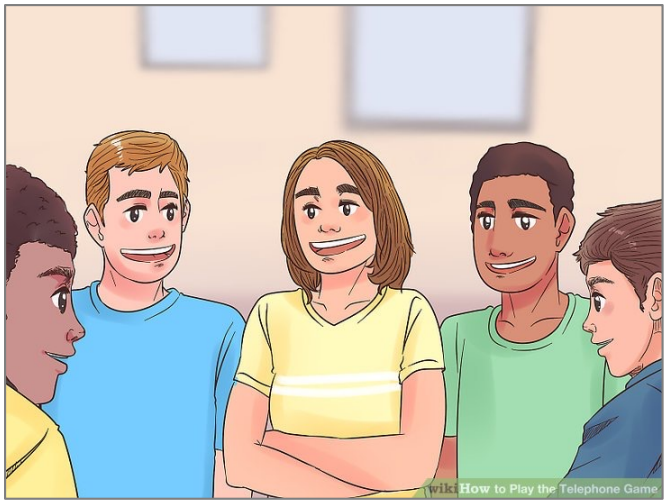
State Handling

CS 4640
Programming Languages for Web Applications

[Based in part on SWE 432 and SWE 632 materials by Jeff Offutt, GMU]
[Robert W. Sebesta, "Programming the World Wide Web]

Remember What I Say



Remember this

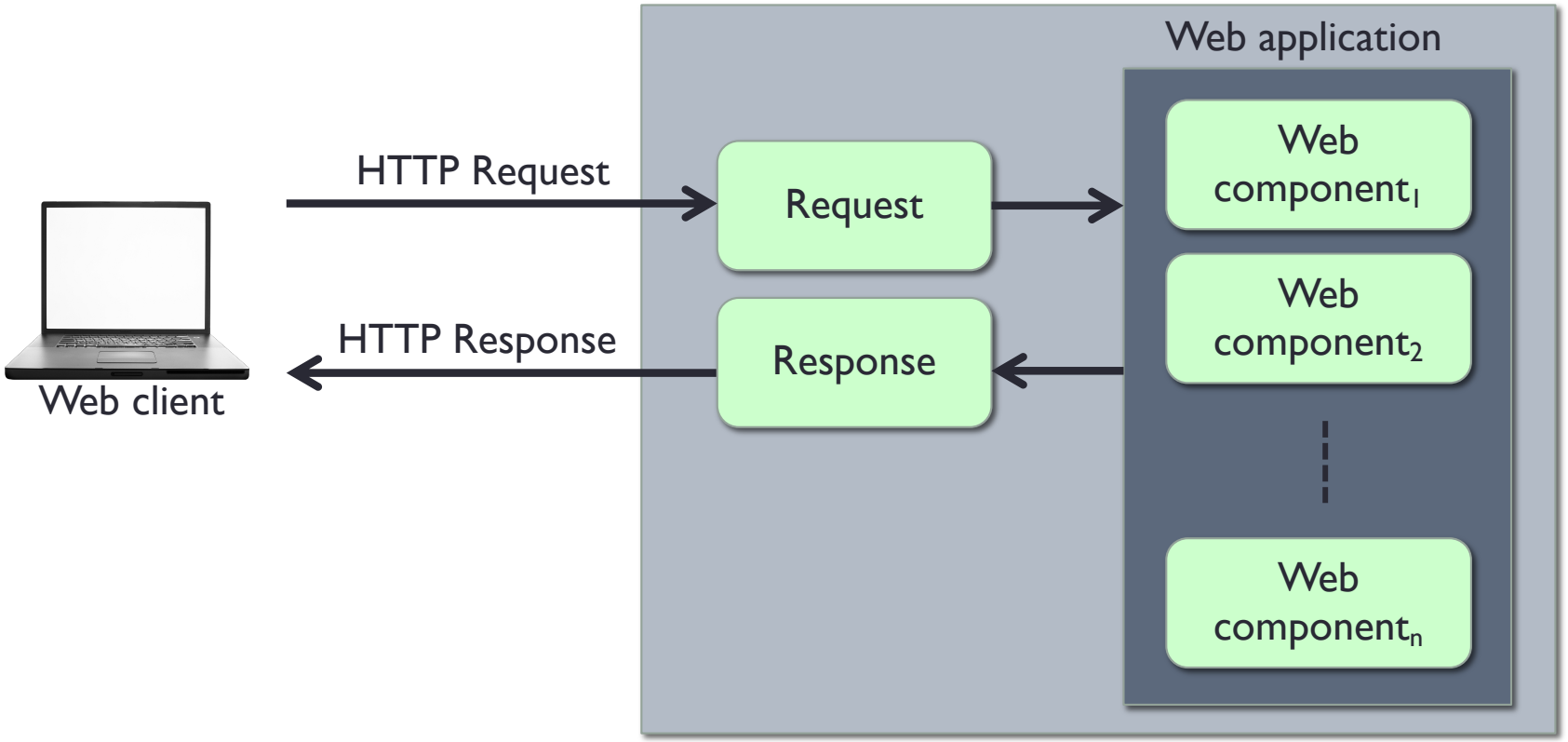
Remember this Really eat

Remember this Really eat Rally car

[images from https://www.wikihow.com/Play-the-Telephone-Game]

revisit

Interactions in Web Apps



Each request/response is treated independently

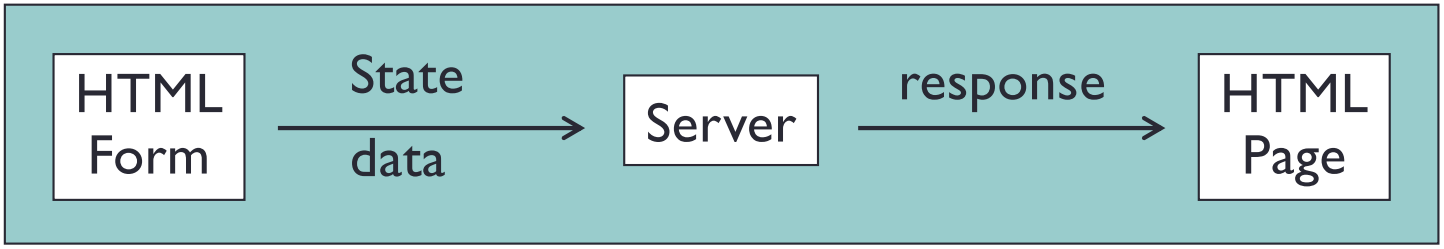
1st interaction → 1st request/response

2nd interaction → 2nd request/response

"Connectionless" ⟶ "Stateless"

Session State Information

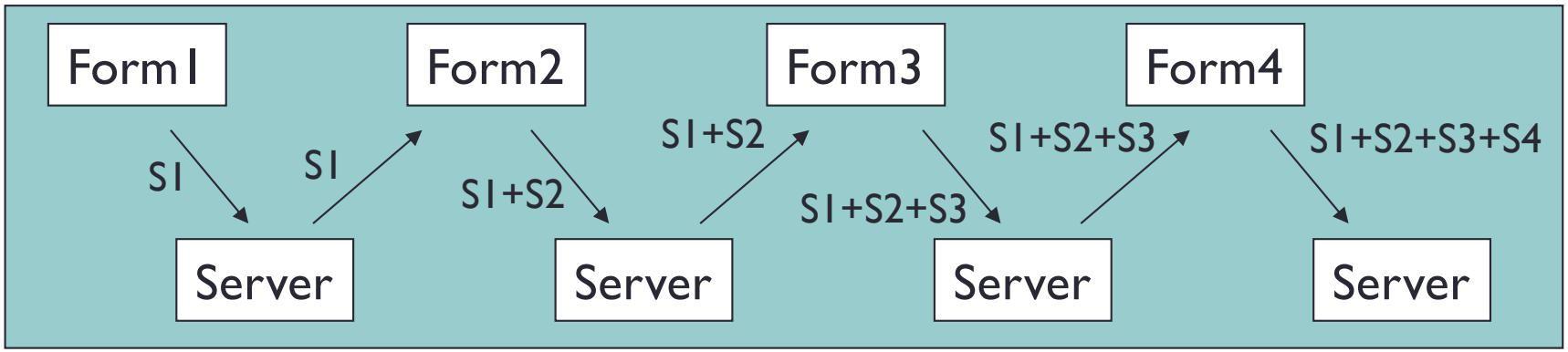
The initial versions of the web suffered from a lack of state



If a web app needed multiple screens, there was no way for state to be accumulated or stored

- This is due to the stateless property of HTTP

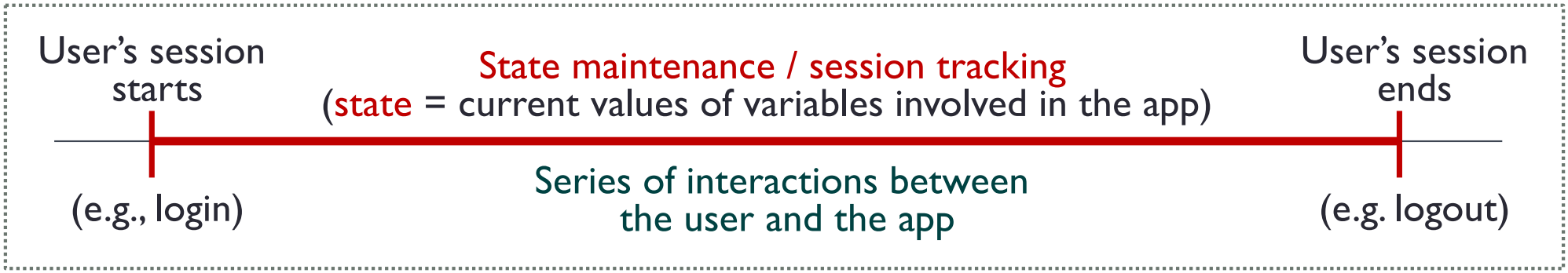
In reality, we want to keep track of the information (i.e., state)



Session Tracking

- Web sites that are service-oriented or e-commerce need to maintain user state – "session tracking"

- User session = a series of related interactions between a client and a web server



- Session tracking = keeping data between multiple HTTP requests

State on the Web

- Unlike standalone / desktop / traditional software:
  - The software components share physical memory
  - The program runs to completion with active memory
- The web brings in unique constraints
  - Connectionless nature of HTTP
  - Distributed software components
- Need ways to store and access variables and objects to keep state in web apps

Public access and parameter passing are not enough for web apps

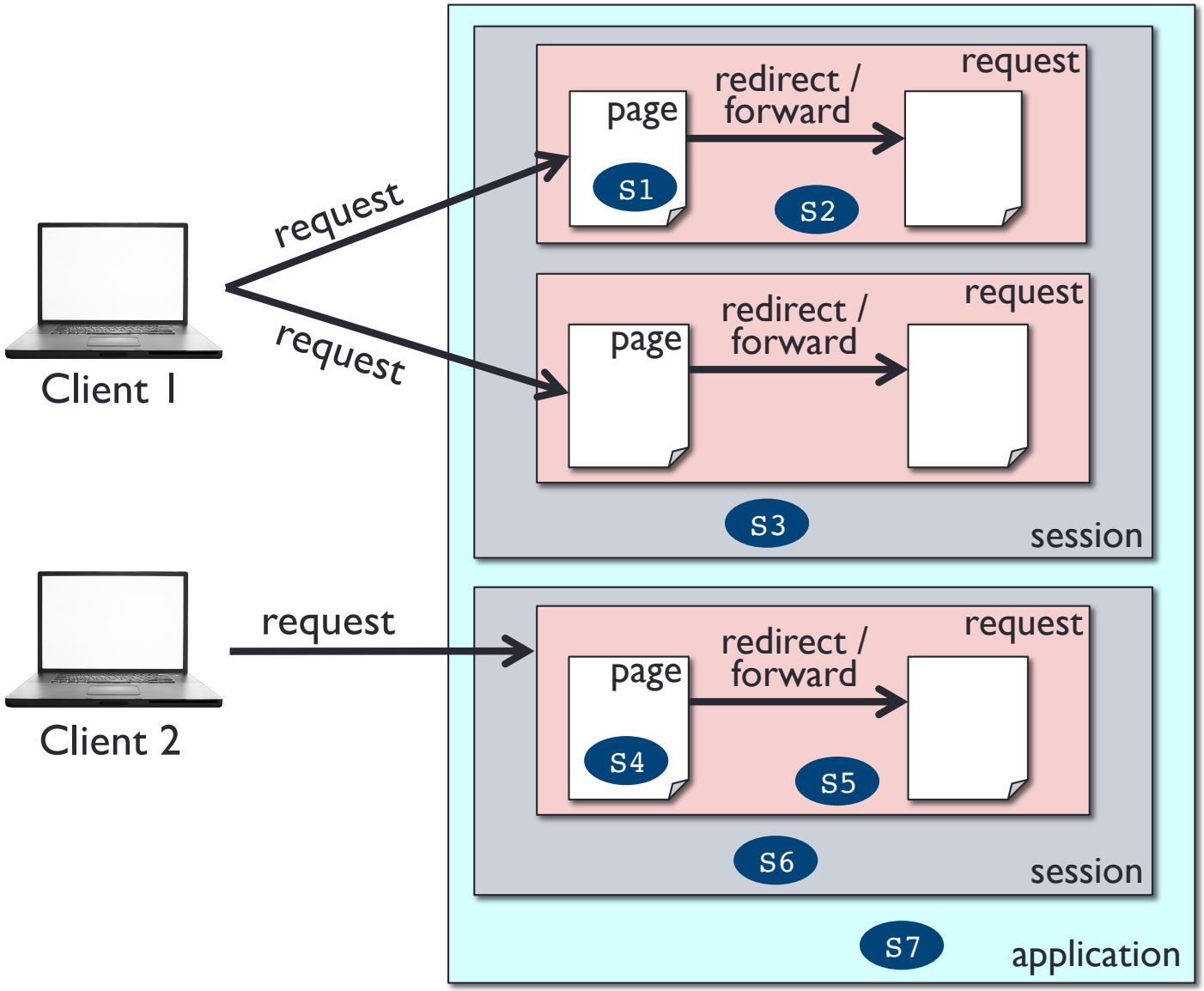
State on the Web

- A web app is composed of several web software components
- Web components do not communicate directly
  - Independent processes (threads)
  - Connectionless protocol
  - Client-server or N-tier architecture
  - Execution flow always goes through a client

How can these independent components share data?

- Session tracking – passing data from one HTTP request to another

Scoping in Web App (in general)

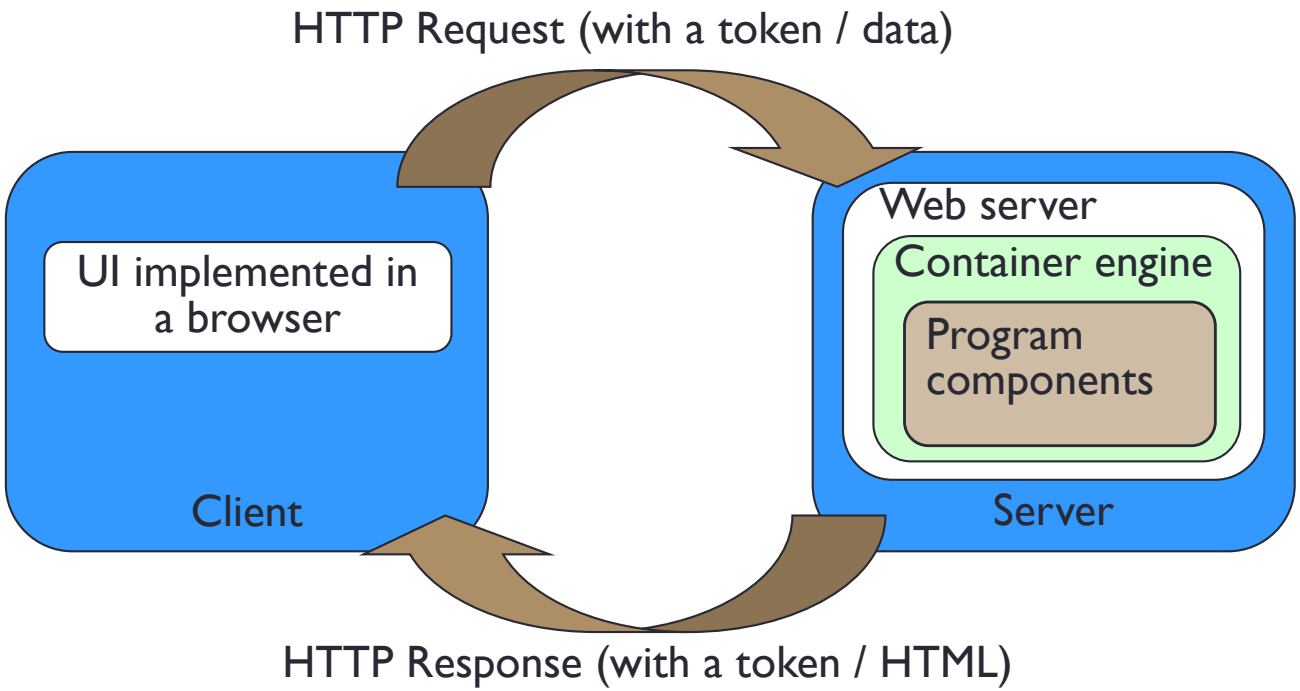


Assume `Si` is a variable (or object) storing the state (i.e., current data)

Session Tracking Methods

- URL rewriting – include data as extra parameters
- Hidden form fields
- Cookies – client-side
- Server-side session objects

All four methods work by exchanging a token between the client and server



URL Rewriting

- Forms usually add parameters

```
URL?P1=v1&P2=v2&P3=v3& ...
```

- You can add value in the URL as parameters

```
href="../filename.php?User=upsorn">
```

- The above example is used as a key to find the saved information about the user **upsorn**

- Drawback
  - Messy and clumsy
  - Long URLs
  - Information on URL is public
  - All HTML pages must be created dynamically

Hidden Form Fields

- Flows of control go through the client
- Store data to be passed from one software component to another in hidden form fields in the HTML
- Generate HTML pages with forms that store "hidden" information

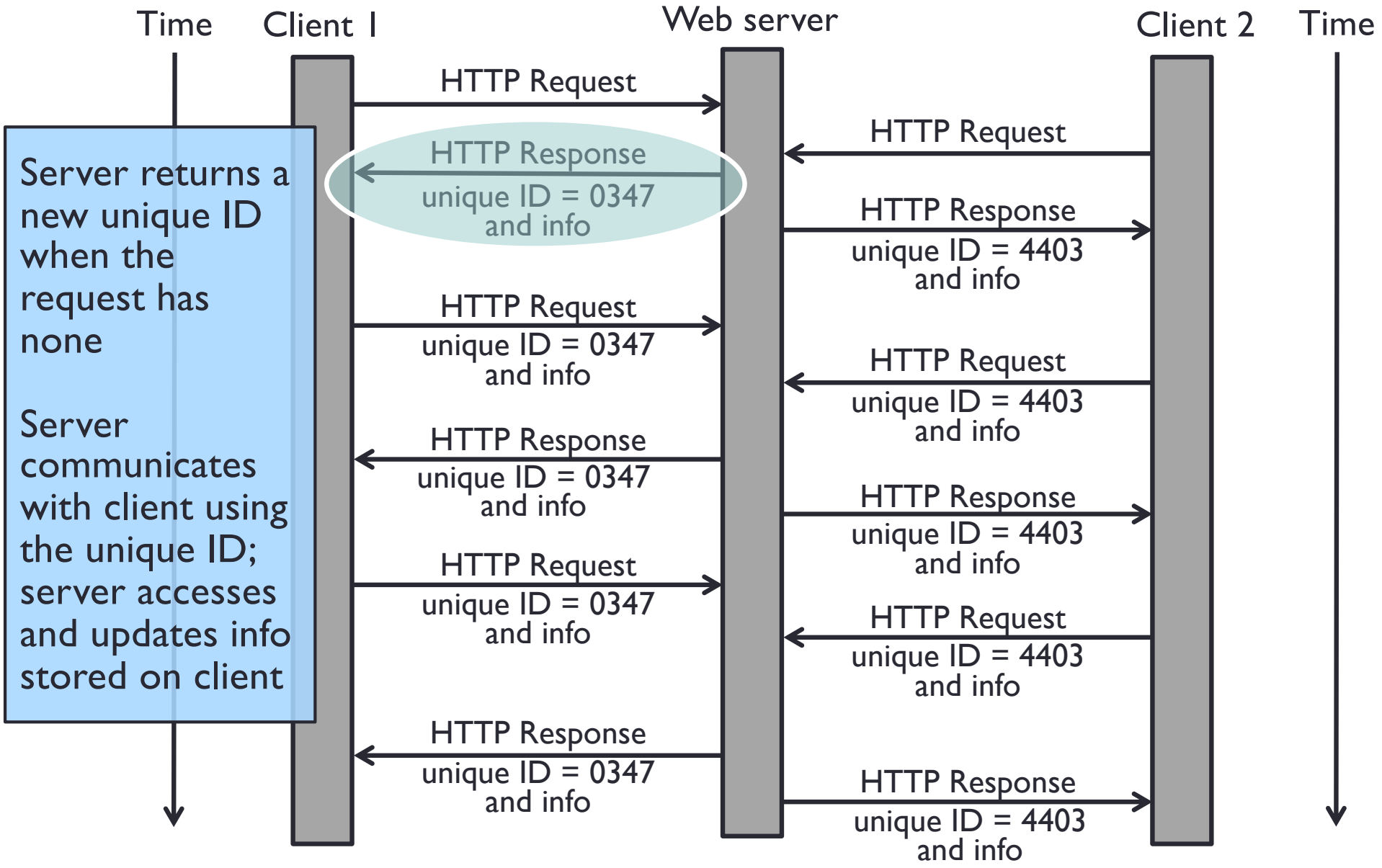
```
<input type="hidden" name="User" value="upsorn" />
```

- Several problems:
  - Insecure – users can see the data
  - Unreliable – users can change the data
  - Undependable – users can use the back button, direct URL entry, and URL rewriting to skip some hidden form fields
- Still useful in limited situations

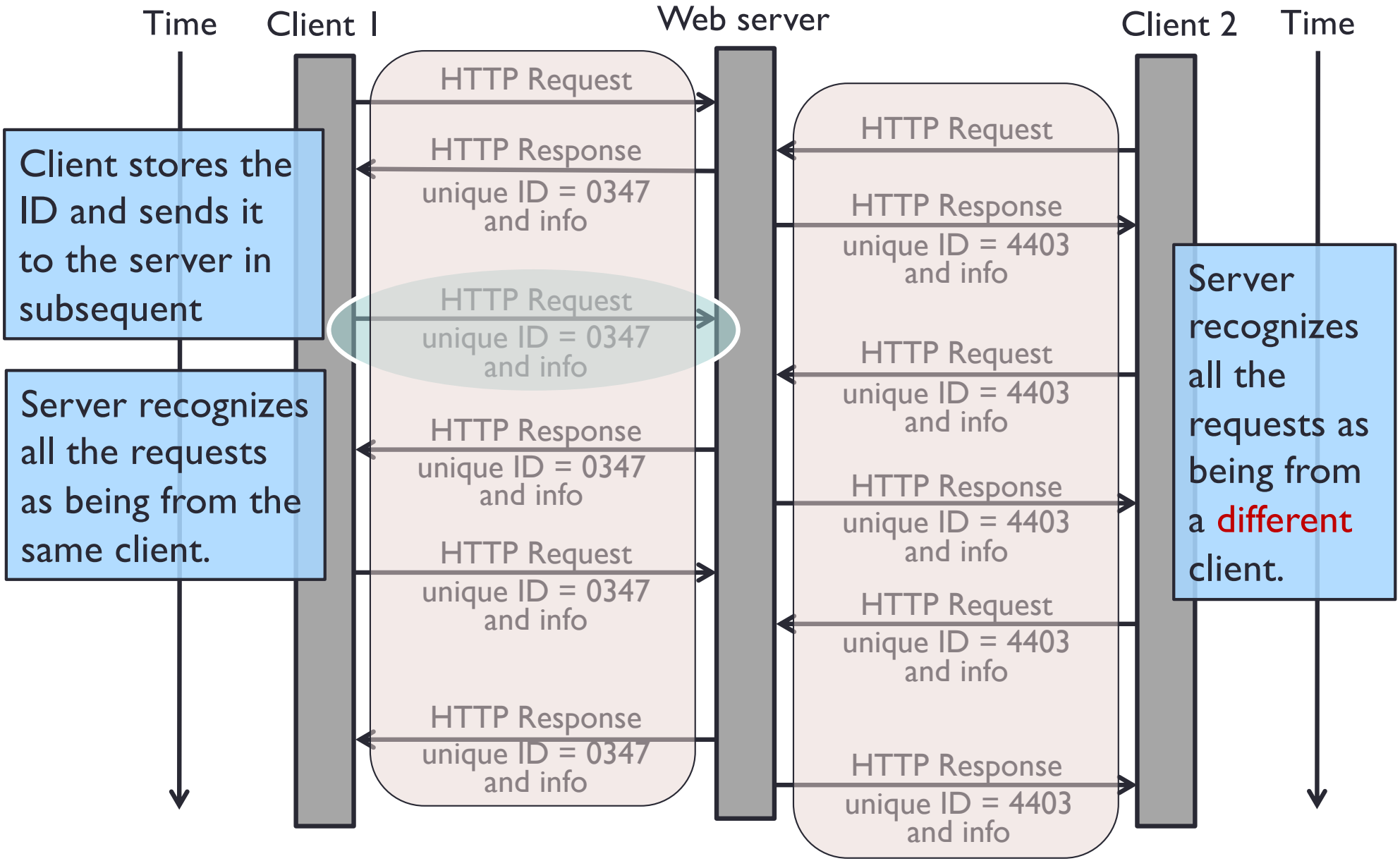
Cookies

- Cookies
  - Small files or text strings
  - Created by the web browser
  - Arbitrary strings stored on the client to keep track of a session
  - Expired after a period of time
  - From the server's (PHP) perspective: val_name = value pairs

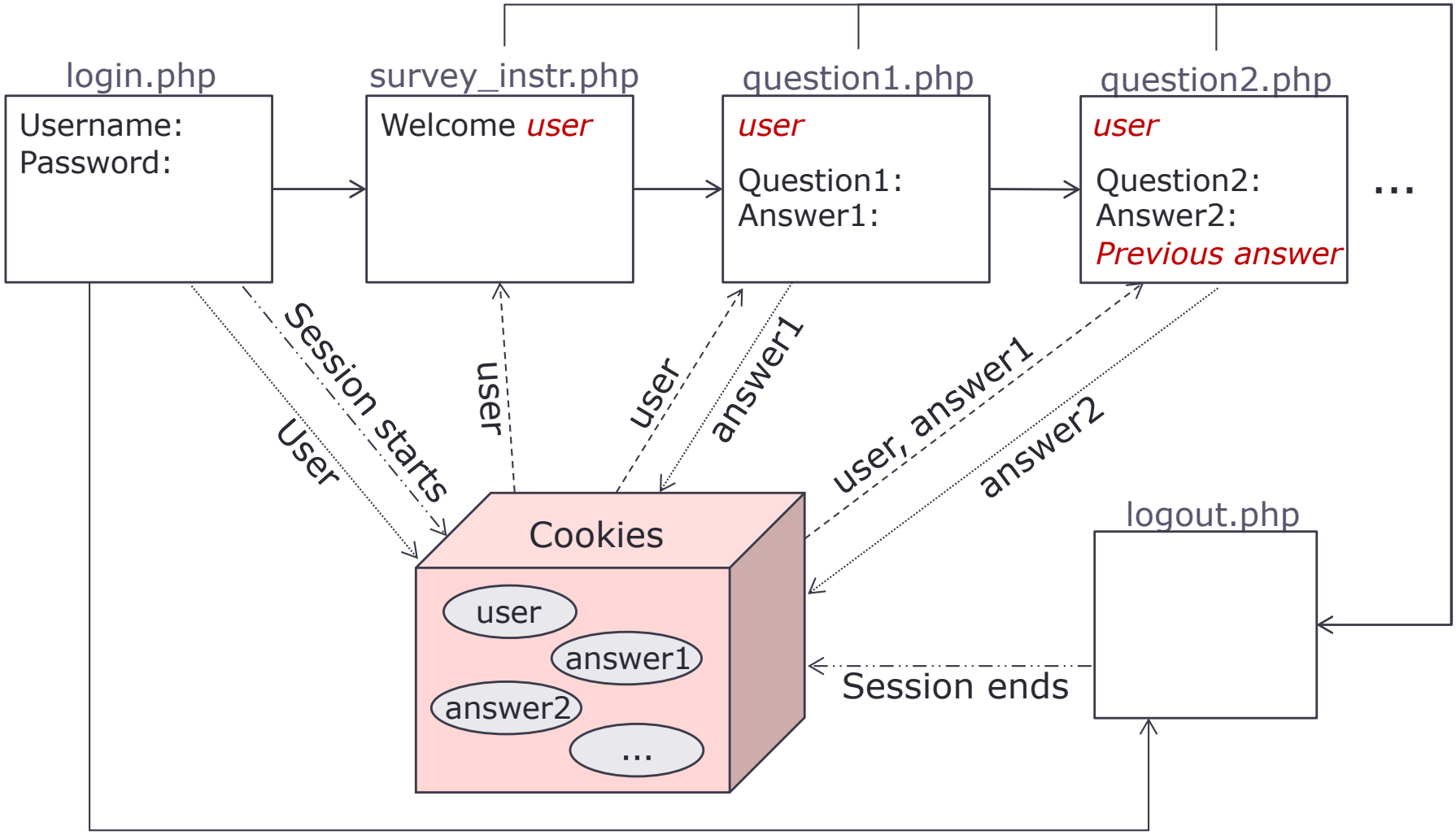
Cookies (Overview)



Cookies (Overview)



Sharing Data with Cookies



[You will implement this later]

Page-centric design

→ Send a request

⇢ Store or update info in Cookies

-·-·→ User's session starts / ends

-----→ Retrieve info from Cookies

Setting Cookies

`setcookie(`*`name, value, expire_time`*`)`

Define a cookie (as name/value pairs) to be sent along with the HTTP header

To set expiration, time() can be used to get the current time in second

`setcookie(`*`name, value,`*` time()+3600)`

Expire in 1 hours (60 * 60 = 3600 seconds)

Getting Cookies

`$_COOKIE[`*`name`*`]`

Once the cookies are set, they are automatically assigned to an implicit $_COOKIE global array variable

To access, use a parameter name as a key

Viewing Cookies

`count($_COOKIE)`

Return the number of param/value pairs stored in cookies

To view state stored in cookies, iterate a $_COOKIE array or specify a parameter name **`$_COOKIE[name]`**

```
foreach ($_COOKIE as $key => $value)
{
    echo "$key maps to $value <br/>";
}
```

Remove Cookies

`unset($_COOKIE[`*`name`*`]`

Delete a param/value pair from cookies
(note: cookie still remains intact in the browser)

To completely remove cookies from the client, set the expiration time to be in the past

`setcookie(`*`name`*`, '', time()–3600);`

Expired an hour ago (60 * 60 = 3600 seconds)

Wrap-Up

- Managing state is fundamental to any program
- Managing state is the most unique aspect of designing and programming web applications
- URL rewriting – messy and insecure
- Hidden form fields – insecure, unreliable, undependable
- Cookies
  - Not visible as part of the HTML content
  - Convenient way to solve a real problem
  - May be disabled by users
  - Cookies go way beyond session tracking – allow behavior tracking

What's next?

- Session tracking with server-side session objects