HERIOT WATT UNIVERSITY

Practical Lab Exercises

Lab – Ajax and JSON

Web Programming (F28WP)

Introduction

In this lab, you'll further develop your understanding of Ajax and JSON.

1.1 Ajax - load() Example

Implement the following sample, with a local client script (i.e., index.html, and on your server, e.g., macs folder (e.g., public_html folder on your university computer, which is accessible from www2.macs.hw.ac.uk/~username/)

Client side (index.html)

```
<html>
   <head>
      <title>TODO supply a title</title>
      <meta charset='UTF-8'>
      <meta name='viewport' content='width=device-width, initial-scale=1.0'>
      <script src='http://code.jquery.com/jquery.js'></script>
      <script>
       $(document).ready(function(){
          setInterval(initTimer, 1000);
       });
       function initTimer(){
          $.ajax({
             url: 'mytimer.php',
             success: function(data) {
                console.log(data);
                 data = data.split(':');
                 $('#hrs').html(data[0]);
                 $('#mins').html(data[1]);
                 $('#secs').html(data[2]);
             }
          });
       }
      </script>
   </head>
   <body>
      <span id='hrs'>0</span>:<span id='mins'>0</span>:<span id='secs'>0</span>
   </body>
</html>
```

Server side (mytimer.php)

```
<?php

header('Content-Type: text/event-stream');
header('Cache-Control: no-cache');

echo date('h:i:s A');
?>
```

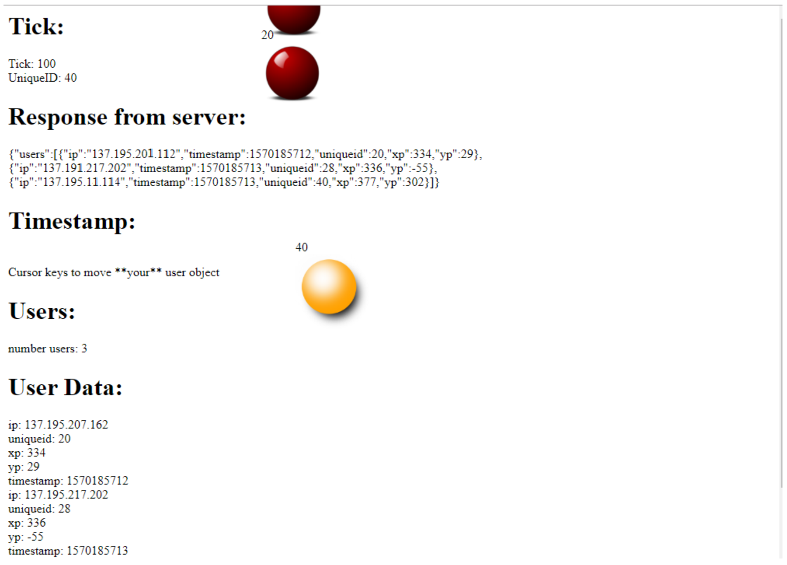
When you run the following script. You should see the content on the page updated (i.e., span section with the information retrieved from mytimer.php).

Try and add extra information, also animations (e.g., move objects around the screen and change the screen background colour dynamically on the fly based on the information received from the php script).

1.1 Server/Client Data Exchange Example (AJAX **long-polling**)

Using a simple php script, you'll update send/receive data between the client-server which is distributed to other connected clients (timing information, tracking identification).

Below shows a simple screenshot of the example (downloadable source code below). Remember, the data.txt file needs write permission. Ideally once you've got it working, you'll move over to a database.



Tick:

Tick: 100
UniqueID: 40

Response from server:

```
{"users":[{"ip":"137.195.201.112","timestamp":1570185712,"uniqueid":20,"xp":334,"yp":29},
{"ip":"137.191.217.202","timestamp":1570185713,"uniqueid":28,"xp":336,"yp":-55},
{"ip":"137.195.11.114","timestamp":1570185713,"uniqueid":40,"xp":377,"yp":302}]}
```

Timestamp:

Cursor keys to move **your** user object

Users:

number users: 3

User Data:

```
ip: 137.195.207.162
uniqueid: 20
xp: 334
yp: 29
timestamp: 1570185712
ip: 137.195.217.202
uniqueid: 28
xp: 336
yp: -55
timestamp: 1570185713
```

Limitations/tasks.

- Currently there is no management (organise/validate users)
- Unique ID is generated on the client (i.e., better for the server to manage this – also a complex unique ID)

- Security flaws (e.g., DOS, validation of data, request to join, ..)
- File data system (e.g., versus a database or shared memory) – improve delays/data read-write thrashing
- All done in a single php script (e.g., split this up, so read/write, as there is no need to save data if you’re only retrieving/accessing latest data)

Download Source code (client.js, index.html, server.php)