

COMP 4021

Lab 8

Overview

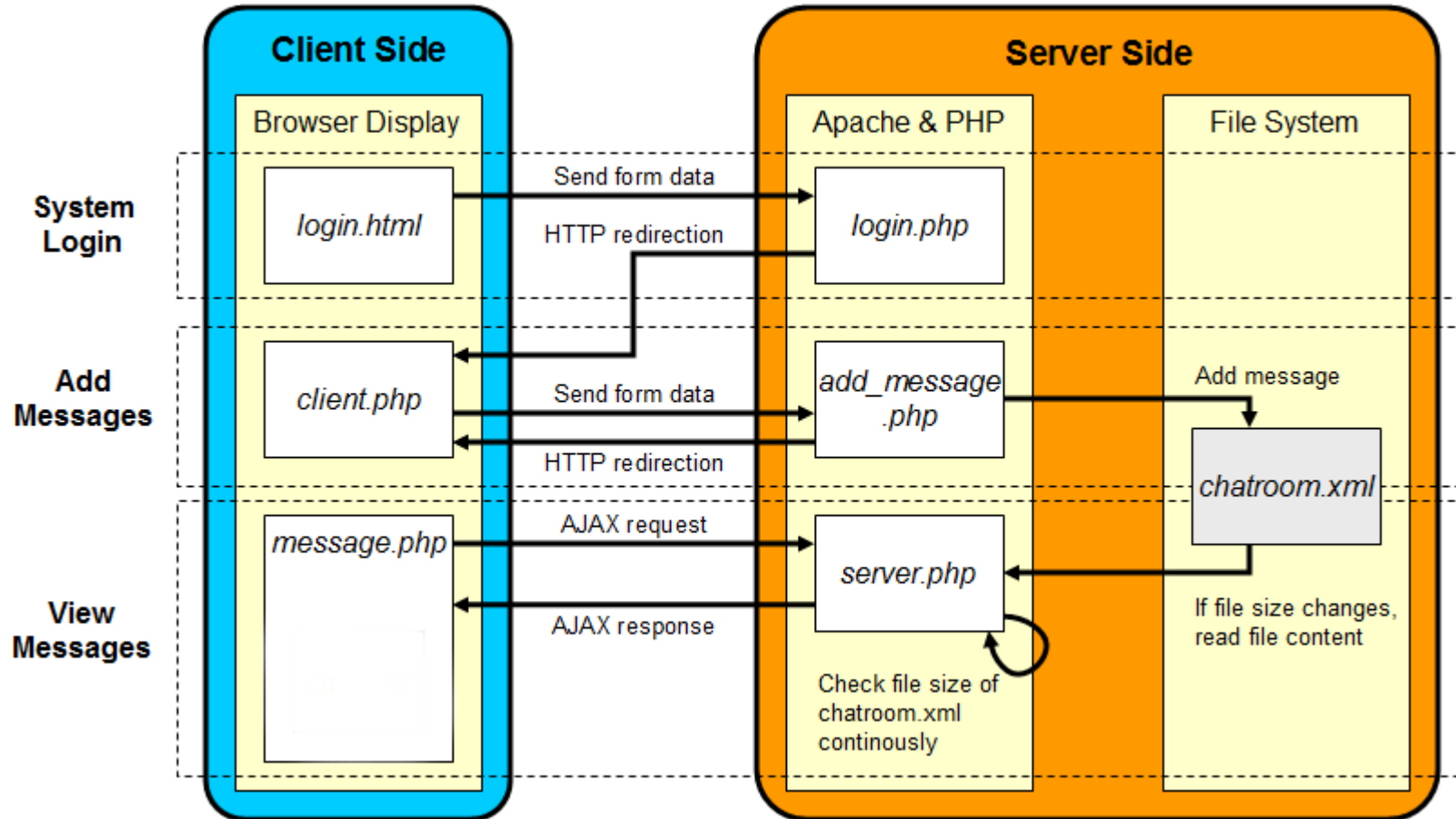
- In this lab, you will further develop your chat system:
 - 1. Complete message.php to display message in the chatting area
 - 2. Select Message color
 - 3. Username Input Checking
- The starting code chatroom.zip is given to you in the previous lab
- Please make sure you have completed the previous lab before you do this one

Clarification

- The starting code given out last time has an extra line in client.php. Remember to delete that, or you may encounter unexpected error(or may not). But now the chatroom.zip files given is updated to the corrected version.

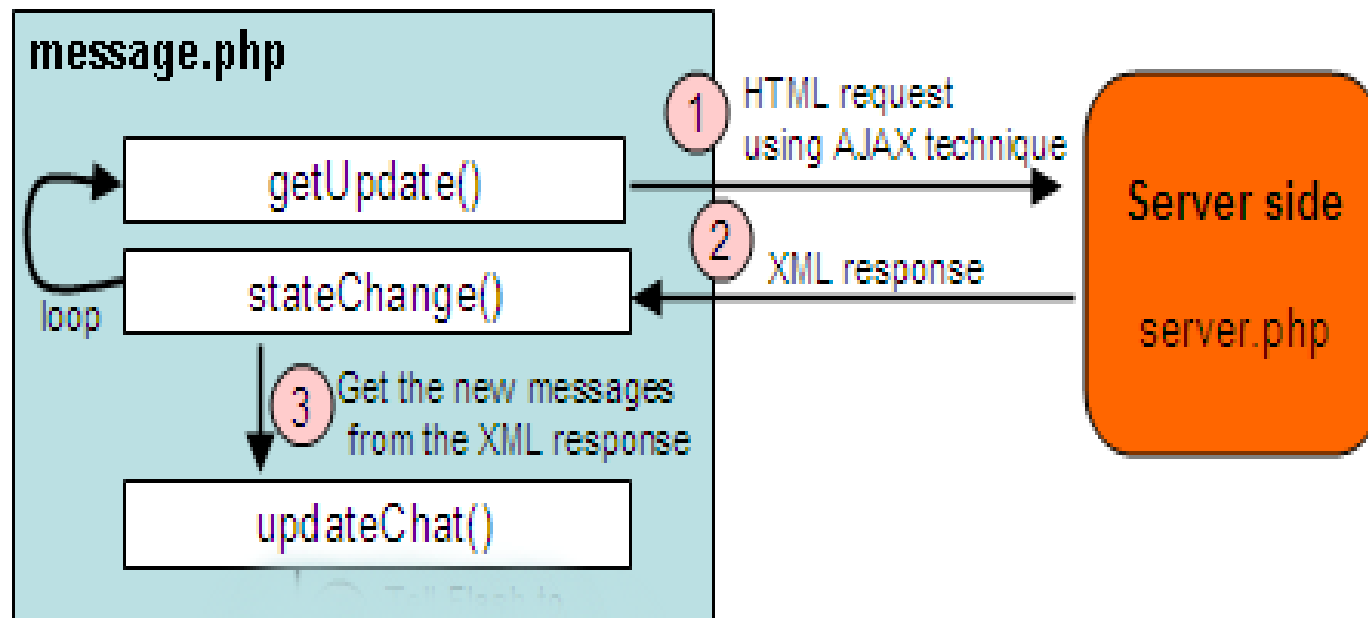
```
function load() {  
    var name = "<?php print $name; ?>";  
  
    //delete this line  
    //window.parent.frames["message"].document.getElementById("username").setAttribute("value", name)  
    |  
    setTimeout("document.getElementById('msg').focus()",100);  
}
```

The System Structure



1. message.php -Getting messages from server

- we use AJAX techniques to handle the request and response, i.e. we send request to the server and process the response using JavaScript
 - **1.1 getUpdate():** create a HTTP request object
 - **1.2 stateChange():** parse XML message to DOM
 - **1.3 updateChat():** Updating the chat area



Using AJAX techniques to get XML data

- AJAX stands for Asynchronous JavaScript and XML
- AJAX is asynchronous which means data is requested from the server and loaded in the background without interfering with the display and behavior of the existing page
- Therefore the entire web page does not have to be reloaded when new data is requested from the server
- `getUpdate()`: send a HTTP request to `server.php`

1.1 getUpdate(): create a HTTP request object

```
//Create a HTTP request object
request =new XMLHttpRequest();

// Set the callback function for this request
request.onreadystatechange = stateChange;

//Set the URL of the file using the POST method
request.open("POST", "server.php", true);

// Set the request header
request.setRequestHeader("Content-type", "application/x-www-form-urlencoded");

// Send the request with the POST data 'datasize=0'
request.send("datasize=0");
```

- The POST data 'datasize=0' specifies that the amount of XML data sent from the server is zero at the beginning

getUpdate():

- Later if we send a request to server.php again to get the latest copy of XML data, the size of the XML data (not the XML data itself) sent from the server previously will be set in the POST data as follows

```
request.send("datasize=" + datasize);
```

- Then the server.php can use the data size to check whether we have got the latest copy of the data on the server
- After sending the request, JavaScript code in message.php will call the callback function if any response is received

1.2 `stateChange()`: parse XML message to DOM

- Because the output from `server.php` is an XML message, we need to parse this XML message and display it appropriately
- We can use a function called `loadXML`

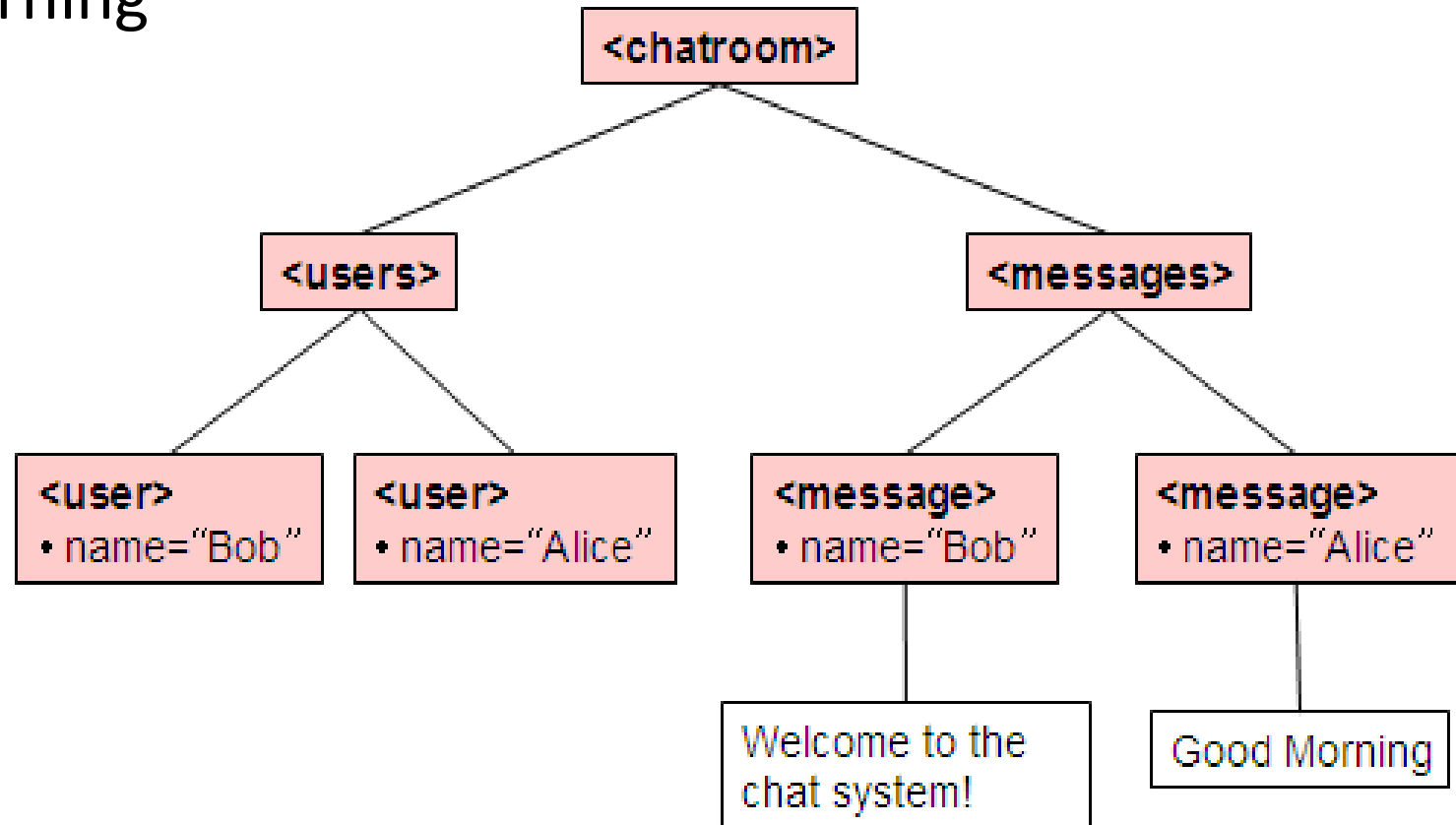
```
xmlDoc = new XMLHttpRequest();  
xmlDoc.loadXML(request.responseText);
```
- This function helps us parse any XML into a nice DOM structure so that we can use normal DOM functions to traverse the data set
- When we receive the response from `server.php`, it will be parsed into DOM document nodes

- For example, if there are two new messages:

Bob: Welcome to the chat system!

Alice: Good Morning

Then the DOM tree:




1.3 `updateChat()`: Updating the chat area

- a for-loop is used to scan through all message nodes inside the DOM tree
- In this lab, you need to add the JavaScript code for updating the chat area.
 - Point to the message nodes
 - Obtain user name and message content from each message node
 - Call function *showMessage()* to display message
 - Record current message node(*messages.length*) so that you can start to process the messages from here next time

2. Select Message Colour

What is your message?

Choose your color: 

- The user can select a colour from **at least 6 colours** in the chat message input form
- When the chat message is sent to the server the message is displayed using the selected colour in the chat room
- The user selects a colour by clicking on a coloured box
- The default colour is the first colour shown in the GUI, e.g. the default colour is black in the above picture

- You can refer to the examples on the course website on how to build the GUI for colour selection
- There are many different ways to do that and one of the easiest ways is to use a collection of divs and a hidden field
- Each message will be displayed using its own colour
- The colour of a chat message is determined by the colour selection of the user on the message input form
- That means you need to store the colour(#rrggbb) in chatroom.xml
- i.e. a new colour attribute should be added to each message element

3. User Name Checking

- Username cannot be empty, or contains any illegal characters
- In login.html, complete the function `checkInput()`
- If username is empty, then alert



- If username contains illegal characters,alert

