A Very Brief Introduction to Web Application Development (III)

Jiun-Long Huang (黃俊龍)
Department of Computer Science
National Chiao Tung University

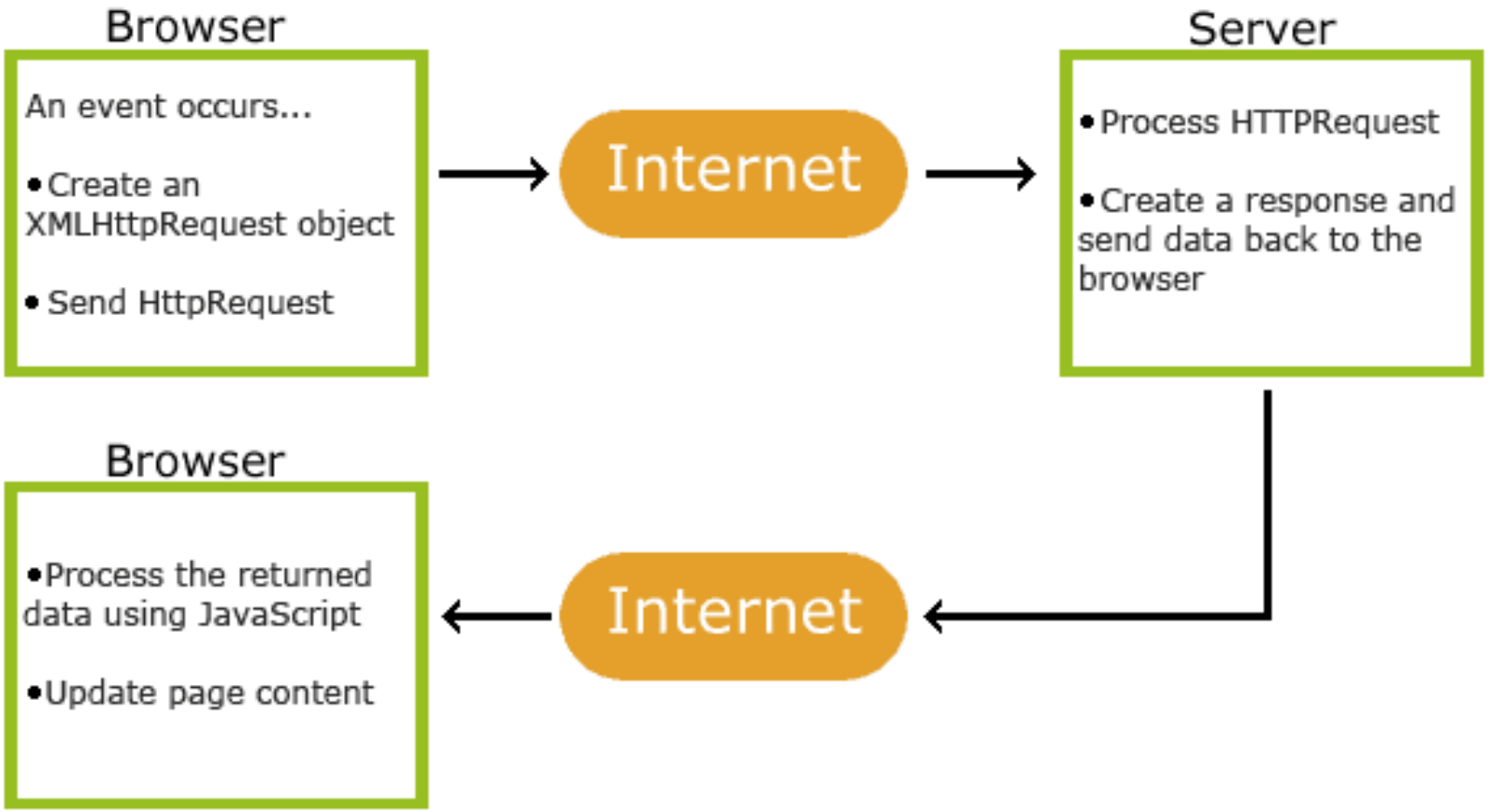
AJAX: Asynchronous JavaScript and XML

Introduction to AJAX

- AJAX is a technique for creating fast and dynamic web pages.
- AJAX allows web pages to be updated asynchronously by exchanging small amounts of data with the server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.
- Classic web pages, (which do not use AJAX) must reload the entire page if the content should change.

XML is not madantory in AJAX

How AJAX Works



XMLHttpRequest

- Create an XMLHttpRequest Object

```
var xhttp = new XMLHttpRequest();
```

- Old Browsers (IE5 and IE6)

```
variable = new ActiveXObject("Microsoft.XMLHTTP");
```

- Example

```
if (window.XMLHttpRequest) {
   // code for modern browsers
   xmlhttp = new XMLHttpRequest();
 } else {
   // code for old IE browsers
   xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
}
```

Method	Description
open(method, url, async)	Specifies the type of request method: the type of request: GET or POST url: the server (file) location async: true (asynchronous) or false (synchronous)
send()	Sends the request to the server (used for GET)
send(string)	Sends the request to the server (used for POST)

Asynchronous Request

- By sending asynchronously, the JavaScript does not have to wait for the server response, but can instead:
  - execute other scripts while waiting for server response
  - deal with the response after the response is ready

Send a Request To a Server

- GET Requests

```
xhttp.open("GET",
           "demo_get2.asp?fname=Henry&lname=Ford",
           true);
xhttp.send();
```

- POST Request

```
xhttp.open("POST", "ajax_test.asp", true);
xhttp.setRequestHeader("Content-type",
    "application/x-www-form-urlencoded");
xhttp.send("fname=Henry&lname=Ford");
```

- With the XMLHttpRequest object you can define a function to be executed when the request receives an answer.

```
xhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
    document.getElementById("demo").innerHTML =
                                        this.responseText;
  }
};
xhttp.open("GET", "ajax_info.txt", true);
xhttp.send();
```

Property	Description
onreadystatechange	Defines a function to be called when the readyState property changes
readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
status	200: "OK" 403: "Forbidden" 404: "Page not found" …
statusText	Returns the status-text (e.g. "OK" or "Not Found")

Synchronous Request

- Since the code will wait for server completion, there is no need for an onreadystatechange function.

```
xhttp.open("GET", "ajax_info.txt", false);
xhttp.send();
document.getElementById("demo").innerHTML =
                                    xhttp.responseText;
```

index_ajax.php

```
<?php
  session_start();
  # remove all session variables
  session_unset();
  # destroy the session
  session_destroy();
  $_SESSION['Authenticated']=false;
?>

<!DOCTYPE html>
<html>
<script>
function check_name(uname)
{
  if (uname!="")
  {
    var xhttp = new XMLHttpRequest();
```

```
  xhttp.onreadystatechange = function() {
    var message;
    if (this.readyState == 4 && this.status == 200) {
      switch(this.responseText) {
        case 'YES':
          message='The user name is available.';
          break;
        case 'NO':
          message='The user name is not available.';
          break;
        default:
          message='Oops. There is something wrong.';
          break;
      }
      document.getElementById("msg").innerHTML = message;
    }
  };
  xhttp.open("POST", "check_name.php", true);
  xhttp.setRequestHeader("Content-type",
                         "application/x-www-form-urlencoded");
  xhttp.send("uname="+uname);
}
```

```
  else
    document.getElementById("msg").innerHTML = "";
}
</script>
<body>
<h1>Login</h1>
<form action="login.php" method="post">
  User Name:
  <input type="text" name="uname"><br>
  Password:
  <input type="password" name="pwd"><br>
  <input type="submit" value="Login">
</form>
<h1>Create Account</h1>
<form action="register.php" method="post">
  User Name:
  <input type="text" name="uname"
      oninput="check_name(this.value);"><label id="msg"></label><br>
  Password:
  <input type="password" name="pwd"><br>
  <input type="submit" value="Create Account">
</form>
</body>
</html>
```

check_name.php

```
<?php
$dbservername='localhost';
$dbname='examdb';
$dbusername='examdb';
$dbpassword='examdb';

try {
  if (!isset($_REQUEST['uname']) || empty($_REQUEST['uname']))
  {
    echo 'FAILED';
    exit();
  }

  $uname=$_REQUEST['uname'];
  $conn = new PDO("mysql:host=$dbservername;dbname=$dbname",
                  $dbusername, $dbpassword);
  # set the PDO error mode to exception
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

  $stmt=$conn->prepare("select username from users where
username=:username");
  $stmt->execute(array('username' => $uname));
```

```
  if ($stmt->rowCount()==0)
  {
    echo 'YES';
  }
  else
  {
    echo 'NO';
  }
}
catch(Exception $e)
{
  echo 'FAILED';
}
?>
```

JSON: JavaScript Object Notation

Introduction to JSON

- JSON stands for JavaScript Object Notation
- JSON is a lightweight data-interchange format
  - Especially compared with XML
- JSON is "self-describing" and easy to understand
- JSON is language independent

Processing JSON by Javascript

- Sending Data

```
var myObj = {name: "John", age: 31, city: "New York"};
var myJSON = JSON.stringify(myObj);
window.location = "demo_json.php?x=" + myJSON;
```

- Receiving Data

```
var myJSON = '{"name":"John", "age":31, "city":"New York"}';
var myObj = JSON.parse(myJSON);
document.getElementById("demo").innerHTML = myObj.name;
```

Processing JSON by PHP

- Decoding JSON Data

```
<!DOCTYPE html>
<html>
<body>

<?php
$jsonobj = '{"Peter":35,"Ben":37,"Joe":43}';

var_dump(json_decode($jsonobj));
?>

</body>
</html>
```

```
object(stdClass)#1 (3)
{ ["Peter"]=> int(35)
["Ben"]=> int(37)
["Joe"]=> int(43) }
```

Processing JSON by PHP

- Encoding object to JSON

```
<!DOCTYPE html>
<html>
<body>

<?php
$age = array("Peter"=>35, "Ben"=>37, "Joe"=>43);

echo json_encode($age);
?>

</body>
</html>
```

```
{"Peter":35,"Ben":37,"Joe":43}
```

References

- HTML Tutorial, https://www.w3schools.com/html/default.asp
- PHP Tutorial, https://www.w3schools.com/php/default.asp
- https://www.w3schools.com
- https://www.php.net