ICT286
Web and Mobile Computing

Topic 10
Arrays, Ajax, jQuery, JSON and Cookies: More Examples

Objectives

- Be able to pass form arrays from the client to the server
- Understand why uri encoding is needed and be able to convert a string to uri encoding in JavaScript.
- Be able to use Ajax to send user input to server using HTTP POST method.
- Be able to use jQuery methods for Ajax, including `load`, `$.get` and `$.post`.
- Be able to retrieve files from the server using jQuery Ajax methods. Be able to send files to server using jQuery Ajax methods.
- Be able to exchange objects using JSON between client and server.
- Understand and be able to handle cookies in PHP.

Example 1: Send Form Arrays to PHP

Initial Script `tuna-get.php`:

```
<html>
<head>
<title> Tuna Cafe </title>
</head>
<body>
   <h3> Welcome to the Tuna Cafe Survey! </h3>
   <form  action="tunaresults-get.php"  method="GET" >
       <?php
       $menu = ['Tuna Casserole', 'Tuna Sandwich', 'Tuna Pie',
                                'Grilled Tuna', 'Tuna Surprise'];
       print 'Please indicate all your favorite dishes.<br />';
       for ($i=0; $i<count($menu); $i++)
          print "<input type=\"checkbox\" name=\"prefer[]\"
value=\"$i\" /> $menu[$i] <br />";
      ?>
      <input type="submit" value="Click To Submit" />
      <input type="reset" value="Erase and Restart" />
   </form>
</body>
</html>
```

Example 1: Send Form Arrays to PHP

- We created the checkboxes using the following PHP code:

```
print "<input type=\"checkbox\" name=\"prefer[]\"
      value=\"$i\" /> $menu[$i]";
```

  Note the double quote characters inside the input element must be escaped.

- The HTML code generated from the above PHP script contains 5 checkboxes inside the form element. These checkboxes are all named `prefer[]` with the corresponding values from 0 to 4.
- These array elements are accessible in JavaScript using the name `prefer[]` (including the square brackets). Eg.,

```
menu = document.getElementsByName("prefer[]");
numMenusSelected = 0;
for (var i=0; i<menu.length; ++i)
   if (menu[i].checked)
       ++numMenuSelected;
```

Example 1: Send Form Arrays to PHP

- When the submit button is clicked, the values of those checkboxes that the user has selected are sent to the server side in the form of an array of the same name: `prefer`.
- For example, if the user clicked the first checkbox (`value="0"`) and third checkbox (`value="2"`), then the following user input data would be sent to the server side:

```
prefer[]=0
prefer[]=2
```

- In the server script, this user input data can be obtained from the `$_GET` (or `$_POST`) array using the array name `prefer`, for example:

```
$preferArray = $_GET["prefer"]
```

Note: unlike in JavaScript, the array name doesn't contain the square brackets.

Example 1: Send Form Arrays to PHP

Server Script: `tunaresults-get.php:`

```
<html> <head><title> Tuna Cafe </title></head>
<body>
<h3> Tuna Cafe Results Received </h3>
<?php
    $menu = ['Tuna Casserole', 'Tuna Sandwich', 'Tuna Pie',
                'Grilled Tuna', 'Tuna Surprise'];
    $preferArray = $_GET[ “prefer” ];
    if (count($preferArray) == 0 ) {
        print ’<br/>Oh no! Please pick something as your favorite!';
    } else {
        print '<br>Your selections were <ul>';
        foreach ($preferArray as $item)
             print "<li>$menu[$item]</li>";
        print '</ul>';
    }
?>
</body>
</html>
```

Example 1: Send Form Arrays to PHP

- This example is available at `Examples/array`. The file names are `tuna-get.php` and `tunaresults-get.php`.
- In the same directory, there is also a similar example, which uses HTTP POST method. The file names are `tuna-post.php` and `tunaresults-post.php`.

Url Encoding

- One of the media types that are used to send data to the server is

  `application/x-xml-form-urlencoded`

- For example, we often use query string to send user input to the server, as in HTTP GET method
- Query string is part of the url. URL cannot contain spaces and some special characters such as quotation marks, and it cannot contain non-ASCII characters either.
- When such characters are present, they must be url encoded.

Url Encoding

- Example

```
var x = 'hong xie " 好';
var y = encodeURI(x);
```

- Variable y contains:

```
Hong%20Xie%20%22%20%E5%A5%BD
```

Example 2: Send User Data With POST Method

```
<body>
  Your name: <input id="name" value="John Doe" /><br/>
    Your age: <input id="age"  value="31" /><br/>
  Where do you live: <input id="city" value="Perth, WA" />
  <br/> <br/>

  <button onclick="sendUserInput()">
       Click to win the prize
  </button> <br/>

    <div id="demo"></div>
</body>
```

Example 2: Send User Data With POST Method

```
function sendUserInput() {
    var data = "name=" +
       encodeURI(document.getElementById('name').value);
    data += "&age=" +
       encodeURI(document.getElementById('age').value);
    data += "&city=" +
       encodeURI(document.getElementById('city').value);
    sendData(data);
}
```

Example 2: Send User Data With POST Method

```
function sendData(data){
   var xhr = new XMLHttpRequest();
   xhr.onreadystatechange=function(){
      if (this.readyState==4 && this.status==200){
         processResponse(this.responseText);
            }
       }
   xhr.open("POST", "getPrize.php", true);
   // without the following header the data
   // cannot be received by the server
   xhr.setRequestHeader("Content-type",
            "application/x-www-form-urlencoded");
     xhr.send(data);
}
```

Example 2: Send User Data With POST Method

```
function processResponse(prize){
   var html = "<p>Dear " +
      document.getElementById('name').value + ",<br/>";
     if (prize>0) {
       html+="Congratulations! You are the lucky winner of \$"
                 +  prize + ".<br/>";
     } else {
       html += "Sorry, you are not the winner.<br/>";
     }
   html+="Best wishes,<br/>Prize Committee, October 2018 </p>";
    document.getElementById('demo').innerHTML = html;
}
```

Example 2: Send User Data With POST Method

Server script: `getPrize.php`:

```
<?php
$name = $_POST["name"];
$age = $_POST["age"];
$city = $_POST["city"];
if ($age >25 && strpos($city, "Perth") !== false) {
     echo 1000;
} else {
     echo 0;
}
?>
```

Note: PHP function `strops` return false if the string "`Perth`" can't be found in variable `$city`.

This example is available at `Examples/ajax/ex6`.

jQuery Ajax Methods

- jQuery provided a number of methods for using Ajax objects that substantially simplify Ajax programming:
- `$( selector ).load( url, data, … );`
  - Load the server response directly into an HTML element
  - May use either GET method if data is provided via a query string
  - Or use POST method if data is provided as a second argument.
- `$.get(url,data,callback(data, status, …), …);`
  - Use GET method to get server response
  - Server response is processed inside the callback
- `$.post(url,data,callback(data, status, …), …);`
  - Use POST method to get server response
  - Server response is processed inside the callback

Example 3: Loading A File From Server

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.
min.js"></script>
<script>
    $(document).ready(function(){
            $('button').click(function(){
// without a selector:
//          $("#demo").load('demo_ajax.txt');
// with a selector
                  $("#demo").load('demo_ajax.txt #p2');
            });
      });
</script>
</head>
```

Example 3: Loading A File From Server

```
<body>

<div id="demo">Test</div>
<button>Change the above</button>
</body>
</html>
```

This example is available at `Examples/jquery/ex1.`

Example 4: Loading a File From Server With a Callback

```
$(document).ready(function(){
    $('button').click(function() {
        $("#demo").load('demo_ajax.txt',
            function(responseText, statusText, xhr) {
                if (statusText == "success")
                    alert("It's a success!");
                else
                    alert("Error: " + xhr.status);

            });
        });
});
```

Example 4: Loading a File From Server With a Callback

```
<body>

<div id="demo">Test</div>
<button>Change the above</button>
</body>
</html>
```

This example is at `Examples/jquery/ex2.`

Example 5: Get Server Response With GET Method

```
<body>

Your name: <input type="text" value="John Doe"/>
<div id="demo"></div>
<button>get greeting</button>
</body>
</html>
```

Example 5: Get Server Response With GET Method

```
$(document).ready(function(){
    $('button').click(function() {
        $("#demo").load('getGreeting.php?name=' +
            encodeURI($('input').val()));
        // without uri encoding, the following woulld
        // not work if the name contains a space
        // $("#demo").load('getGreeting.php?name=' +
        //       $('input').val());
    });
});
```

This load method uses GET method as data is provided in the form of a query string.

Example 5: Get Server Response With GET Method

File `getGreeting.php`:

```
<?php
  $name = $_GET['name'];
// test whether $().load loads stuff asynchronously
// sleep(10);
  if ($name !== "")
      echo "Hello, ", $name, "!";
  else
      echo "Who are you?";
?>
```

This complete example is at `Examples/jquery/ex3`.

Example 6: Get Server Response With POST Method

```
<body>

Name: <input id="name" type="text" value="John Doe"/>
<br/>
City: <input id="city" type="text" value="Perth, WA"/>
<div id="demo"></div>
<button>get greeting</button>
</body>
</html>
```

Example 6: Get Server Response With POST Method

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/
jquery.min.js"></script>
<script>
    $(document).ready(function(){
        $('button').click(function() {
            $("#demo").load('getGreeting.php',
{ name: $('#name').val(), city: $('#city').val() } );
        });
    });
</script>
</head>
```

Example 6: Get Server Response With POST Method

```
<?php

  $name = $_POST['name'];
  $city = $_POST['city'];
  if ($name !== "")
      echo "Hello, ", $name, " of ", $city, "!";
  else
      echo "Who are you?";

?>
```

This complete example is at `Examples/jquery/ex4`.

Example 7: Get Server Response With $.get Method

```
<body>

Name: <input id="name" type="text" value="John Doe"/> <br/>
City: <input id="city" type="text" value="Perth, WA"/>

<div id="demo"></div>
<button>get greeting</button>

</body>
</html>
```

Example 7: Get Server Response With $.get Method

```
$(document).ready(function(){
   $('button').click(function() {
      $.get('getGreeting.php',
            { name: $('#name').val(),
              city: $('#city').val() },
      // the following also works even though there
      // is no uri encoding
      // $.get('getGreeting.php?name='
      //    + $('#name').val()
      //    + "&city=" + $('#city').val(),
            function(data, status) {
                $('#demo').html(data);
      });
   });
});
```

Example 7: Get Server Response With $.get Method

File `getGreeting.php`:

```
<?php
  $name = $_GET['name'];
  $city = $_GET['city'];
  if ($name !== "")
      echo "Hello, ", $name, " of ", $city, "!";
  else
      echo "Who are you?";
?>
```

This complete example is at `Examples/jquery/ex5`.

Example 8: Get Server Response With $.post Method

```
<body>

Name: <input type="text" id="name" value="John Doe" /> <br/>
City: <input type="text" id="city" value="Perth, WA" />

<div id="demo"></div>
<button>get greeting</button>

</body>
</html>
```

Example 8: Get Server Response With $.post Method

```
<script>
    $(document).ready(function(){
        $('button').click(function() {
            $.post('getGreeting.php',
                { name: $('input').val(),
                  city: $('#city').val() },
                function(data, status){
                   $('#demo').html(data);
            });
        });
    });
</script>
```

Example 8: Get Server Response With $.post Method

File `getGreeting.php`:

```
<?php
  $name = $_POST['name'];
  $city = $_POST['city'];
  if ($name !== "")
      echo "Hello, ", $name, " of ", $city, "!";
  else
      echo "Who are you?";
?>
```

This complete example is at `Examples/jquery/ex6`.

Exchange Data Between Client And Server Using JSON

- Under HTTP protocol, data exchange between client and server must be in text.
- Complex data such as JavaScript objects and PHP objects can be converted to JSON notation which is a text string before data transfer.
- In JavaScript,
  - function `JSON.stringify( obj )` converts a JavaScript object into a JSON string
  - function `JSON.parse ( json )` converts a JSON string into a JavaScript object
- In PHP,
  - function `json_encode( $obj )` converts PHP object into a JSON string
  - function `json_decode( $json )` converts a JSON string into a PHP object

Example 9: JavaScript Object And JSON

```
<h3> From JSON String to JS Object</h3>
Name: <input id="name"/> <br/>
Age: <input id="age"/> <br/>
City: <input id="city"/> <br/>
<script>
    // note the key and value in JSON string
    // must be double quoted!
    var jsonString = '{"name" : "John", "age" : 31,
"city" : "Perth"}';
    myObj = JSON.parse(jsonString);
    $('#name').val(myObj.name);
    $('#age').val(myObj.age);
    $('#city').val(myObj.city);
</script>
```

Example 9 JavaScript Object And JSON

```
<h3> From JS Object to JSON String</h3>
<div id="demo"></div>

<script>
    var myObj = { name: "John", age: 31, city: "Perth"};
    var myJSON = JSON.stringify(myObj);
    $('#demo').html(myJSON);
</script>
```

This complete example is at `Examples/json/ex1`.

Example 10: More On JavaScript Object And JSON

```
<h3> In JS Object, a value can be a string, a number, null, or true/false </h3>
<div id="demo1"></div>

<script>
    var myObj = { "name": "John", "age": 31, "city": null, "enrolled": true};
    var myJSON = JSON.stringify(myObj);
    $('#demo1').html(myJSON);
</script>
```

Example 10: More On JavaScript Object And JSON

```
<h3> In JS Object, a key can quoted or not quoted, a string value is either double quoted or single quoted
</h3>
<div id="demo2"></div>

<script>
    var myObj = { name: 'John', "age": 31, 'city': null, enrolled: true};
    var myJSON = JSON.stringify(myObj);
    $('#demo2').html(myJSON);
</script>
```

Example 10: More On JavaScript Object And JSON

```
<h3> In a JSON String, both key and value must be double quoted  </h3>
Name: <input id="name"/> <br/>
Age: <input id="age"/> <br/>
City: <input id="city"/> <br/>
Enrolled: <input id="enrolled"/> <br/>
```

Example 10: More On JavaScript Object And JSON

```
<script>
    var jsonString = '{"name" : "John", "age" : 31,
"city" : null, "enrolled": true}';
// the following does not work:
//    var jsonString = "{'name' : 'John', 'age' : 31,
'city' : null, 'enrolled': true}";
    myObj = JSON.parse(jsonString);
    $('#name').val(myObj.name);
    $('#age').val(myObj.age);
    $('#city').val(myObj.city);
    $('#enrolled').val(myObj.enrolled);
</script>
```

This complete example is at `Examples/json/ex2`.

Example 11: Complex JavaScript Object And JSON

```
<h3> From complex JS Object to JSON String</h3>
<div id="demo"></div>
<script>
    var myObj = {
        name: "John",
        age: 31,
        cities: {
            city1: "Perth",
            city2: "Sydney",
            city3: "New York"
        }
    };
    var myJSON = JSON.stringify(myObj);
    $('#demo').html(myJSON);
</script>
```

Example 11: Complex JavaScript Object And JSON

```
<h3> Fromn complex JSON String to JS Object</h3>
Name: <input id="name"/> <br/>
Age: <input id="age"/> <br/>
City: <input id="city1"/> <br/>
City: <input id="city2"/> <br/>
City: <input id="city3"/> <br/>
```

Example 11: Complex JavaScript Object And JSON

```
<script>
    var jsonString = '{"name" : "John", "age" : 31,
"cities" : {"city1": "Perth", "city2": "Sydney",
"city3": "New York"}}';
    myObj = JSON.parse(jsonString);
    $('#name').val(myObj.name);
    $('#age').val(myObj.age);
    $('#city1').val(myObj.cities["city1"]);
    $('#city2').val(myObj["cities"]['city2']);
    $('#city3').val(myObj['cities'].city3);
</script>
```

This complete example is at `Examples/json/ex3`.

Example 12: Retrieving A JSON File Using $.get Method

```
$(document).ready(function(){
    $('button').click(function() {
        $.get('person.json', function(myObj, status) {
            // myObj is a JS object converted from JSON
            if (status == "success") {
                $('#name').val(myObj.name);
                $('#age').val(myObj.age);
                $('#city').val(myObj.city);
                $('#mobile').val(myObj.mobile);
                $('#enrolled').val(myObj.enrolled);
            }
        });
    });
});
```

Example 12: Retrieving A JSON File Using $.get Method

```
<body>

Name: <input id="name"/> <br/>
Age: <input id="age"/> <br/>
City: <input id="city"/> <br/>
mobile: <input id="mobile"/> <br/>
Enrolled: <input id="enrolled"/> <br/>
<br/>
<button>load JSON</button>

</body>
</html>
```

Example 12: Retrieving A JSON File Using $.get Method

File `person.json`:

```
{ "name" : "John",
  "age" : 31,
  "city" : "Perth",
  "mobile": null,
  "enrolled" : false
}
```

This complete example is at `Examples/json/ex4`.

Example 13: Post Data To A JSON File On Server

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/
jquery.min.js"></script>
<script>
    $(document).ready(function(){
        $('button').click(function() {
            var myObj = { name:$('#name').val(),
                          age:$('#age').val(),
                          city:$('#city').val(),
                          mobile:$('#mobile').val(),
                          enrolled:$('#enrolled').val()
                        }
```

Example 13: Post Data To A JSON File On Server

```
            $.post('person.php',
                { person:JSON.stringify(myObj) },
                function(data, status) {
                    if (status == "success") {
                        $('#op').val("successful");
                    }
            });
        });
    });
</script>
</head>
```

Example 13: Post Data To A JSON File On Server

```
<body>

Name: <input id="name"/> <br/>
Age: <input id="age"/> <br/>
City: <input id="city"/> <br/>
mobile: <input id="mobile"/> <br/>
Enrolled: <input id="enrolled"/> <br/>
<br/>
Operation: <input id="op" /> <br/>

<button>Post personal details</button>

</body>
</html>
```

Example 13: Post Data To A JSON File On Server

File `person.php`:

```
<?php
    $myJSON= $_POST['person'];
    $fd = fopen("person.json", "w")
        or die("unable to open open file person.json");
    fputs($fd, $myJSON)
        or die("unable to write to file person.json");
    fclose($fd);
?>
```

This complete example is at `Examples/json/ex5`.

Example 14 GET Data From A JSON File On Server

```
$(document).ready(function(){
    $('button').click(function() {
        $.get('person.php', function(data, status) {
            if (status == "success") {
                var myObj = JSON.parse(data);
                $('#name').val(myObj.name);
                $('#age').val(myObj.age);
                $('#city').val(myObj.city);
                $('#mobile').val(myObj.mobile);
                $('#enrolled').val(myObj.enrolled);
                $('#op').val("successful");
             }
        });
    });
});
```

Example 14: GET Data From A JSON File On Server

```
<body>

Name: <input id="name"/> <br/>
Age: <input id="age"/> <br/>
City: <input id="city"/> <br/>
mobile: <input id="mobile"/> <br/>
Enrolled: <input id="enrolled"/> <br/>
<br/>
Operation: <input id="op" /> <br/>

<button>Get personal details</button>

</body>
</html>
```

Example 14: GET Data From A JSON File On Server

File `person.php`:

```
<?php
    $fd = fopen("person.json", "r")
        or die("unable to open open file person.json");
    $myJSON = fgets($fd)
        or die("unable to write to file person.json");
    fclose($fd);
    echo $myJSON;
?>
```

This complete example is at `Examples/json/ex6`.

Example 15: Creating JSON in PHP

- The final example shows how to create JSON using three different PHP types:
  - PHP objects
  - PHP sequential arrays
  - PHP associative arrays
- The complete example can be found in

  `Examples/json/ex9`

Example 15: Creating JSON in PHP

File `getJSON.php`:

```
<?php
// PHP object
$myObj = new stdClass();
$myObj->name = "John";
$myObj->age = 31;
$myObj->city = "Perth";
$myJSON_obj = json_encode($myObj);

// PHP array
$myArray = array("John", 31, "Perth");
$myJSON_array = json_encode($myArray);
```

Example 15: Creating JSON in PHP

```
// PHP associative array
$myTable = array("name"=>"John", "age"=>31,
"city"=>"Perth");
$myJSON_table = json_encode($myTable);

$type = $_GET["type"];
//$type = "obj";
//$type = "array";
//$type = "table";
```

Example 15: Creating JSON in PHP

```
switch ($type) {
case "obj" :
    echo $myJSON_obj;
    break;
case "array" :
    echo $myJSON_array;
    break;
default :
    echo $myJSON_table;
}

?>
```

Example 15: Creating JSON in PHP

```
<body>
    Name: <input id="name"/> <br/>
    Age: <input id="age"/> <br/>
    City: <input id="city"/> <br/>

    <p>Get JSON from Server using Ajax </p>
    <button onclick="loadDoc('obj');">
         get JSON from PHP Object</button> <br/>
    <button onclick="loadDoc('array');">
         get JSON from PHP Array</button> <br/>
    <button onclick="loadDoc('table');">
         get  JSON from PHP Associative Array</button>
<br/>
</body>
</html>
```

Example 15: Creating JSON in PHP

```
function loadDoc(type){
   var xhr = new XMLHttpRequest();
   xhr.onreadystatechange = function(){
      if (this.readyState == 4 && this.status == 200) {
          fillForm(type, this.responseText);
      }
   }
   xhr.open("GET", "getJSON.php?type="+type, true);
   xhr.send();
}
```

Example 15: Creating JSON in PHP

```
function fillForm(type, json){
   var myObj = JSON.parse(json);
   if (type == "array") {
      document.getElementById('name').value = myObj[0];
      document.getElementById('age').value = myObj[1];
      document.getElementById('city').value = myObj[2];
   } else {
      document.getElementById('name').value =
myObj.name;
      document.getElementById('age').value = myObj.age;
      document.getElementById('city').value =
myObj.city;
   }
}
```

HTTP Cookies

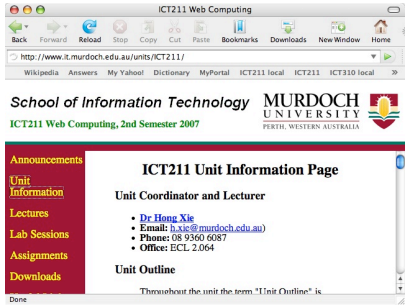
- The HTTP protocol is stateless. In HTTP, all transactions are independent.
  - HTTP requests and responses do not inherently contain information about previous requests/responses.
  - The newer HTTP/1.1 allows for “persistent connections”, but that is for low level TCP connection management.
- However, servers want to keep information about a client, to be used later.
  - E.g., a user’s preference, or previous transactions in electronic shopping, etc.
- Cookies were created to achieve this.

HTTP Cookies

- Cookies were originally proposed by Netscape, and later adopted by other browsers.
- Cookies are sent by servers to clients (browsers), and are managed and stored locally by the clients. Clients send cookie data back to the same server at later sessions.
- Clients have the ability to reject cookies sent by servers.

Example Cookie Transaction (first time)

Browser



1. User requests a resource at www.blah.com

www.blah.com's web server

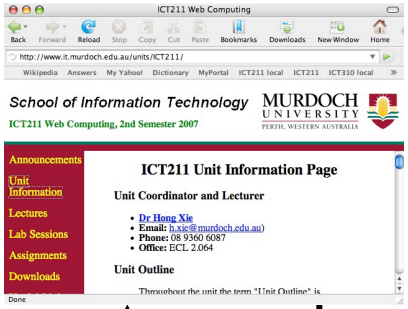
HTTP/1.1 200 OK
Set-Cookie: NAME=lastaccess; DATA=20181015;
…

2. Web server constructs a cookie, by putting the info in the header of the response and sends it back to the browser.

3. Browser stores the cookie "lastaccess:20181015" in a file.

Example Cookie Transaction (subsequent times)

Browser



1. User requests a resource at www.blah.com again

2. Browser sees that there is a local cookie file for www.blah.com, so sends it with the request.

3. Server (or the requested resource) uses the cookie data as appropriate.

4. Web server sends the new cookie data back.

5. Browser stores the new cookie info in the same file as before.

```
GET  /index.html  HTTP/1.1
Cookie: NAME=lastaccess; DATA=20181015;
…
```

www.blah.com's web server

```
HTTP/1.1 200 OK
Set-Cookie: NAME=lastaccess; DATA=20181016;
…
```

The Cookie Myth

- A common misconception about cookies is that it allows servers to get information about you and your machine beyond what it is supposed to.
- Servers can only store in cookies the data which it already has received through an HTTP request, so basically you've already given the server the information.
    - You should be careful what information you actively supply originally, rather than about the cookie.

Support for Cookies

- PHP also has extensions to support manipulating HTTP Cookies.
- Remember:

  Cookies are sent by servers to clients (browsers), and are managed and stored locally by the clients. Servers request these cookies at later sessions.

The Web Application's Tasks for Handling Cookies

- The web application (at the web server side) has to
  - construct the cookie header in HTTP responses
  - read the cookie header in HTTP requests
  - process the data in the cookies.
- PHP extensions for cookies make these steps transparent and easy to use.
- Cookies must be sent from the server to the client prior to any other information, otherwise an error will occur.

Setting Cookies

- A cookie is created and sent using the `setcookie()` function:

```
setcookie(name, value, expiration);
```

  - For example; `setcookie('User', 'Hong');` will send a cookie named ‘User’ to the browser. The value of in the cookie will be ‘Hong’.
  - Names of cookies should not contain any spaces or punctuations and are case-sensitive.
  - Different browsers can deal with cookies in different ways, so ensure you test on different browsers.
  - Cookies must be created before any HTML is created by the script.

Accessing Cookies

- The value in a cookie is accessed via the superglobal array `$_COOKIE` using the name of the cookie:

```
$_COOKIE[name]
```

  - For example; `$_COOKIE['User']` will give the value in the cookie created on the previous slide; i.e., '`Hong`'.
  - You can use the `isSet()` function to determine if the cookie exists; for example:

```
If (isset($_COOKIE['User']))  { // some code
```

  - To delete a cookie use `setcookie()` with just the name of the cookie as a parameter.

Example: Creating a Cookie

```
<?php
      $today = date("d-m-Y H:i:s");
      setcookie ('lastaccess', $today, time()+3600);
?>

<html>
 <body>
  <?php
      $lastaccess = $_COOKIE['lastaccess'];
      if ($lastaccess) {
       echo "<p>Welcome back. " ;
       echo "Your last visit was on $lastaccess.</p>" ;
      } else
       echo "<p>Welcome to this page for the first time.</p>" ;
  ?>
 <body>
</html>
```

The format of $today is "day-month-year hour:minutes:second". Eg "15-10-2018 09:10:07"

The cookie is set with the name "lastaccess", the value of $today, and expires 1 hour (3600 seconds) from when it is created.

testcookie.php

References

- W3Schools Tutorials:
  https://www.w3schools.com

- jQuery Manual:
  http://api.jquery.com/

- PHP Manual:
  http://php.net/manual/en/index.php