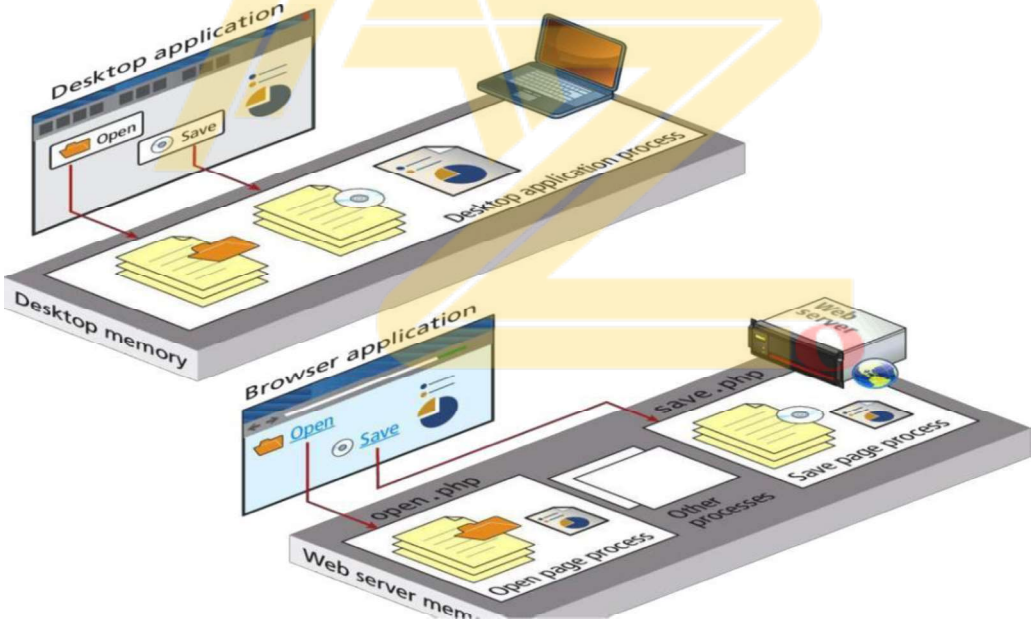
MODULE – 5

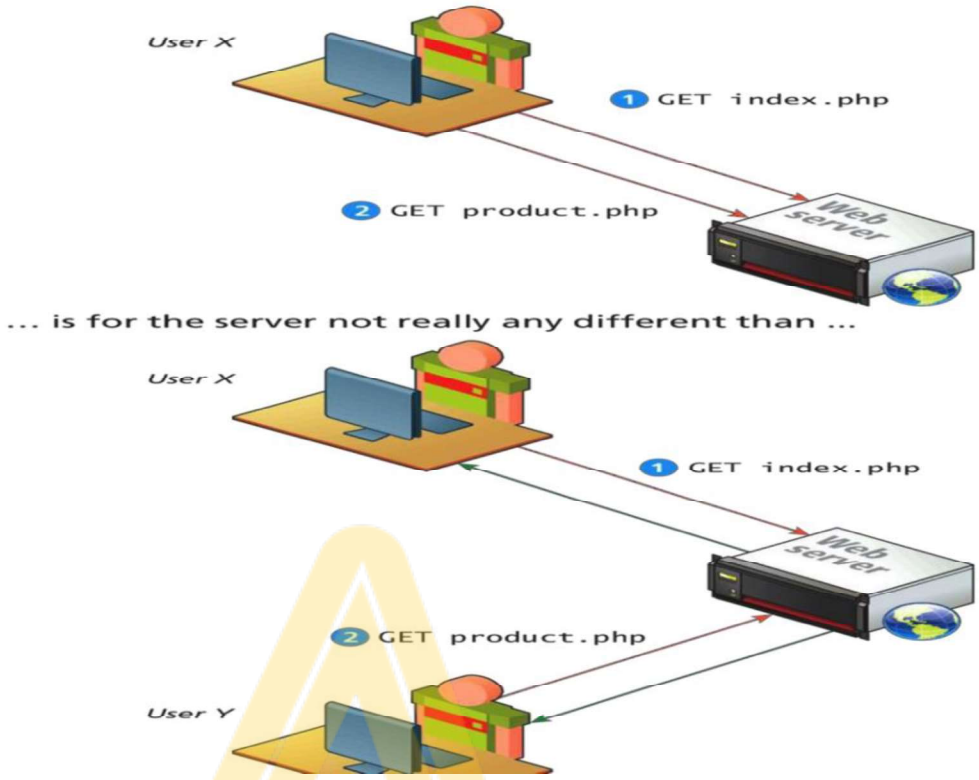
MANAGING STATES

THE PROBLEM OF STATE IN WEB APPLICATIONS

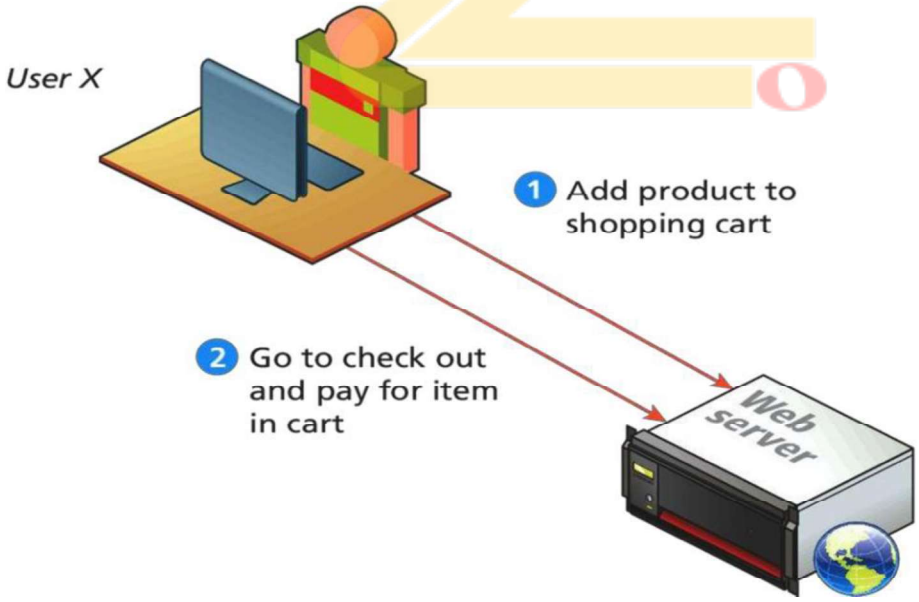
- Until now we have seen how to process user inputs, output information and read & write from other storage media.
- But here we will be examining a development problem that is unique to the world of web development.
- Single user desktop applications do not have this challenge at all because the program information for the user is stored in memory (or in external storage) and thus can be easily accesses throughout the applications.
  - Remember Web applications differ from desktop applications
- Unlike the unified single process that is the typical desktop application, a web application consists of a series of disconnected HTTP requests to a web server where each request for a server page is essentially a request to run a separate program as in below fig



- The web server sees only requests.
- The HTTP protocol does not, without program intervention, distinguish two requests by one source from two requests from two different sources, as in below figure



- There are many occasions when we want the web server to connect requests together.
- Consider the scenario of a web shopping cart, as in below fig.
- In such a case, the user(website owner) most certainly wants the server to recognize that the request to add an item to the cart and the subsequent request to check out and pay for the item in the cart are connected to the same individual



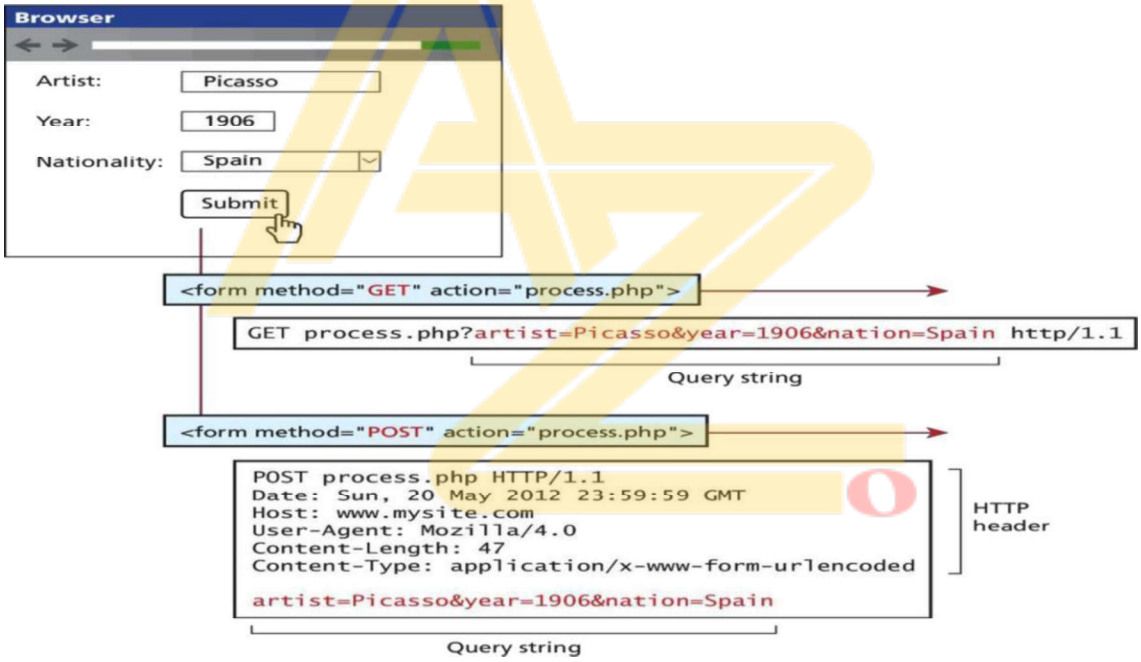
- The rest of the chapter we will explain how web developers and web development environments work together through the constraints of HTTP to solve this particular problem.
- How does one web page pass information to another page?
- What mechanisms are available within HTTP to pass information to the server in our requests?

In HTTP, we can pass information using:

- Query strings
- Cookies

PASSINGINFORMATION VIA QUERY STRINGS

- A web page can pass query string information from the browser to the server using one of the two methods:
  - A query string within the URL(GET)
  - A query string within the HTTP header(POST)



PASSINGINFORMATION VIA THE URL PATH

- Passing information from one page to another is done by query strings but they have drawback.
- The URLs that result can be long and complicated.
- while there is some dispute about whether dynamic URLs(i.e., ones with query string parameters) or static URLs are better from a search engine result optimization (or SEO)

- Dynamic URLs (i.e., query string parameters) are a pretty essential part of web application development.
- ✓ How can we do without them?
- The answer is to rewrite the dynamic URL into a static one (and vice versa). This process is commonly called **URL rewriting**.
- In below fig, the top four commerce – related results for the search term "Reproduction Raphael portrait Donna velata" are shown along with their URLs.
- Notice how the top three do not use query string parameters but instead put the relevant information within the folder path or the file name.

URL rewriting

- We can try doing our own rewriting. Let us begin with the following URL with its query string information:

www.somedomain.com/DisplayArtist.php?artist=16

- One typical alternate approach would be to rewrite the URL to:

www.somedomain.com/artists/16.php

- Notice that the query string name and value have been turned into path names. One could improve this to make it more SEO friendly using the following:

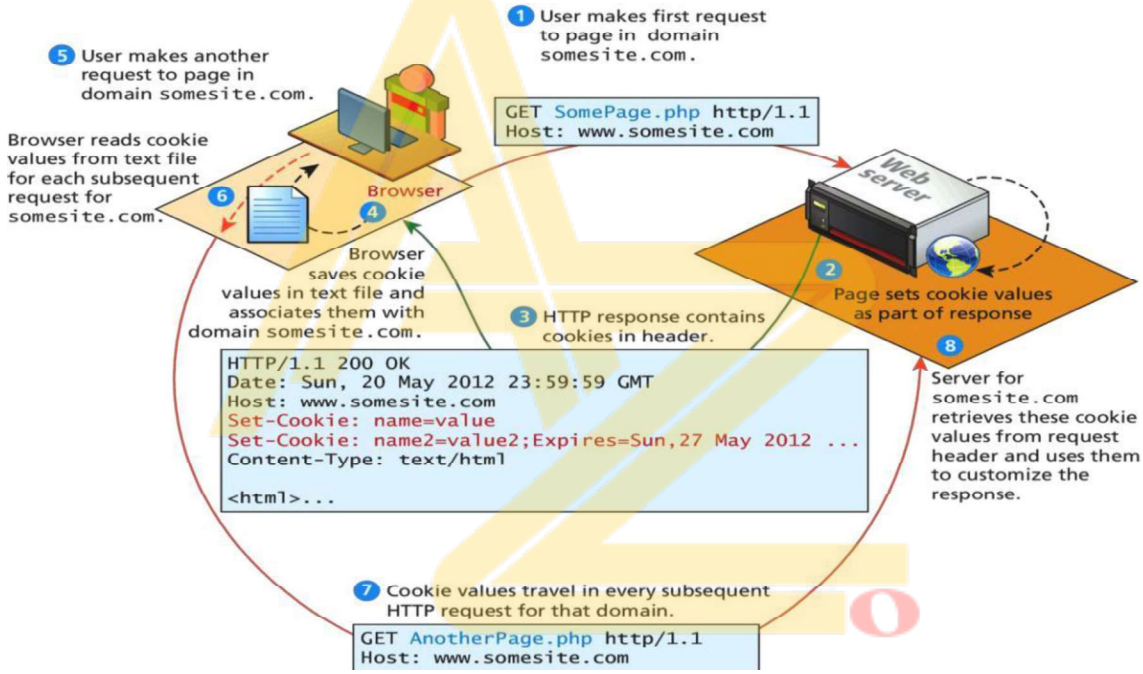
www.somedomain.com/artists/Mary-Cassatt

URL rewriting in Apache and linux

- The mod_rewrite module uses a rule-based rewriting engine that utilizes Perl compatible regular expressions to change the URLs so that the requested URL can be mapped or redirected to another URL internally.

COOKIES

- **Cookies** are a client-side approach for persisting state information.
- They are name=value pairs that are saved within one or more text files that are managed by the browser.
- While cookie information is stored and retrieved by the browser, the information in a cookie travels within the HTTP header.
- Sites that use cookies should not depend on their availability for critical features
- The user can delete cookies or tamper with them



Two kinds of Cookie

- A **session cookie** has no expiry stated and thus will be deleted at the end of the user browsing session.
- **Persistent cookies** have an expiry date specified;

```
<?php
    // add 1 day to the current time for expiry time
    $expiryTime = time()+60*60*24;

    // create a persistent cookie
    $name = "Username";
    $value = "Ricardo";
    setcookie($name, $value, $expiryTime);
?>
```

LISTING 13.1 Writing a cookie

```php
<?php
   if( !isset($_COOKIE['Username']) ) {
      //no valid cookie found
   }
   else {
      echo "The username retrieved from the cookie is:";
      echo $_COOKIE['Username'];
   }
?>
```

LISTING 13.2 Reading a cookie

- In addition to being used to track authenticated users and shopping carts, cookies can implement:
- "Remember me" persistent cookie
- Store user preferences
- Track a user"s browsing behavior

SERIALIZATION

- **Serialization** is the process of taking a complicated object and reducing it down to zeros and ones for either storage or transmission.
- In PHP objects can easily be reduced down to a binary string using the **serialize ()** function.
- The string can be reconstituted back into an object using the **unserialize()** method

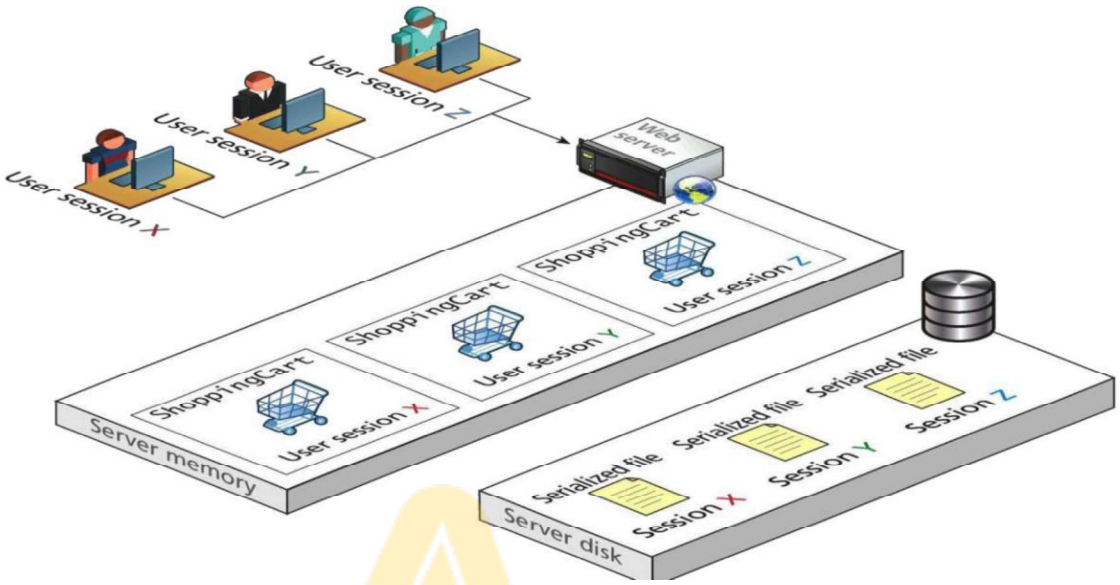
```php
interface Serializable {
   /* Methods */
   public function serialize();
   public function unserialize($serialized);
}
```

LISTING 13.3 The Serializable interface

Application of Serialization

- Since each request from the user requires objects to be reconstituted, using serialization to store and retrieve objects can be a rapid way to maintain state between requests.
- At the end of a request you store the state in a serialized form, and then the next request would begin by deserializing it to re establish the previous state.

SESSIONSTATE



- All modern web development environments provide some type of session state mechanism.
- **Session state** is a server-based state mechanism that lets web applications store and retrieve objects of any type for each unique user session.
- Session state is ideal for storing more complex objects or data structures that are associated with a user session.
- In PHP, session state is available to the via the $_SESSION variable
- Must use session_start() to enable sessions.

```php
<?php

session_start();

if ( isset($_SESSION['user']) ) {
    // User is logged in
}
else {
    // No one is logged in (guest)
}
?>
```

LISTING 13.5 Accessing session state

```php
<?php
include_once("ShoppingCart.class.php");

session_start();

// always check for existence of session object before accessing it
if ( !isset($_SESSION["Cart"]) ) {
   //session variables can be strings, arrays, or objects, but
   // smaller is better
   $_SESSION["Cart"] = new ShoppingCart();
}
$cart = $_SESSION["Cart"];
?>
```

LISTING 13.6 Checking session existence

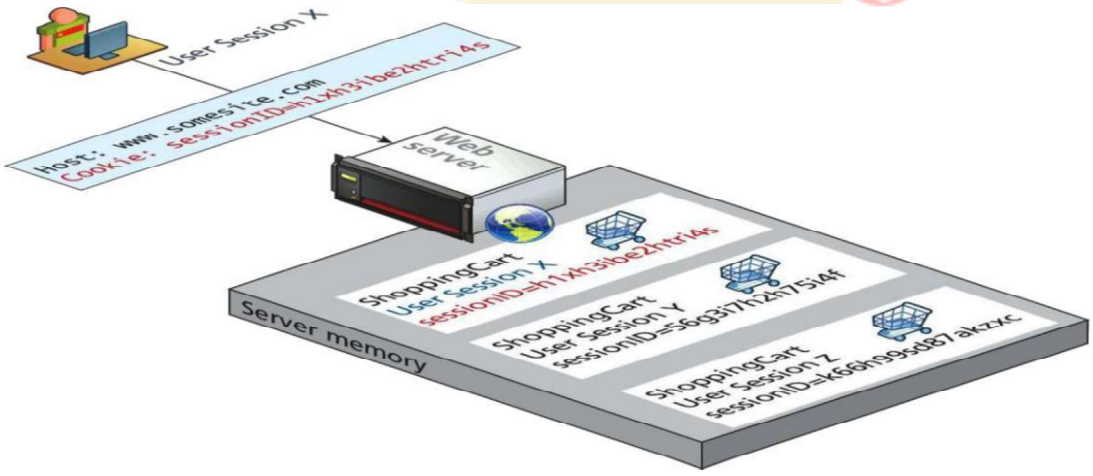

```php
<?php
include_once("ShoppingCart.class.php");

session_start();

// always check for existence of session object before accessing it
if ( !isset($_SESSION["Cart"]) ) {
   //session variables can be strings, arrays, or objects, but
   // smaller is better
   $_SESSION["Cart"] = new ShoppingCart();
}
$cart = $_SESSION["Cart"];
?>
```

LISTING 13.6 Checking session existence

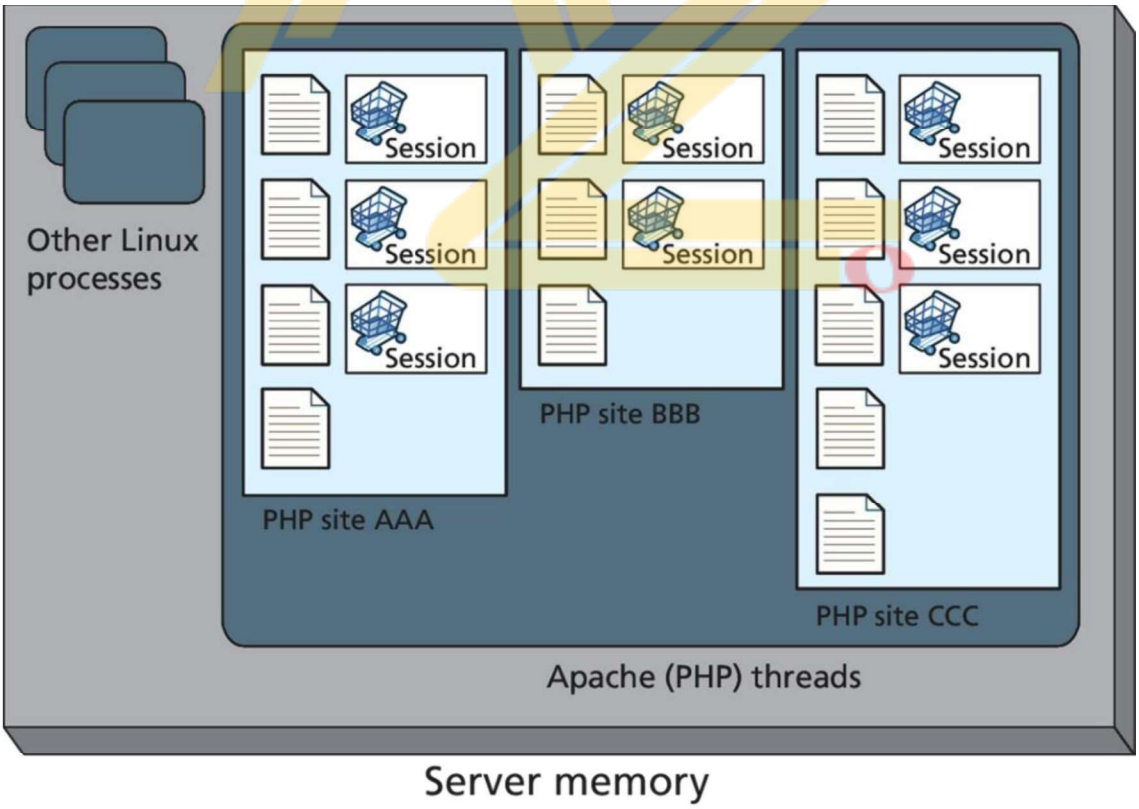
How does state session work?



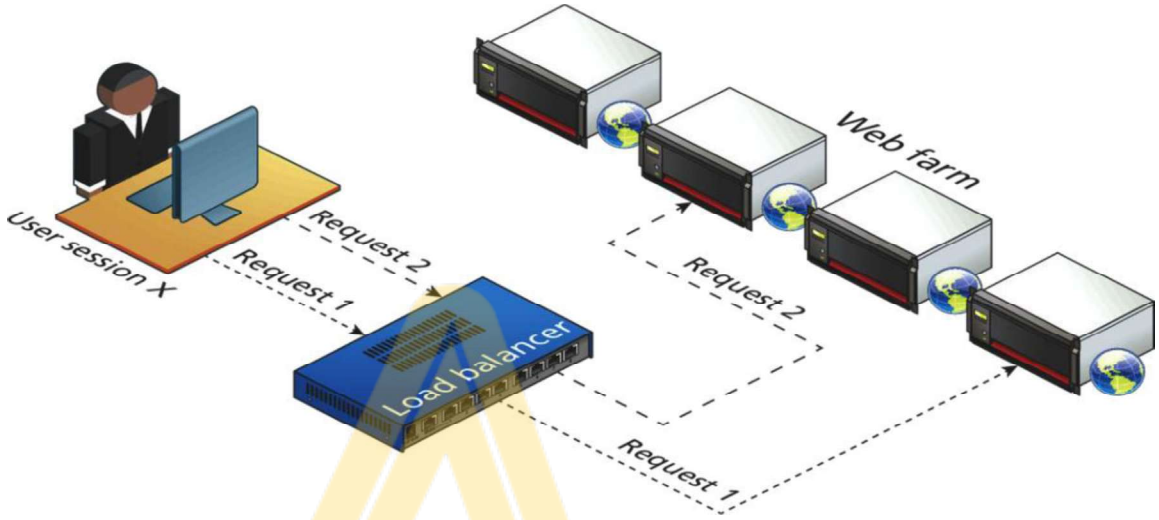
- Sessions in PHP are identified with a unique 32- byte session ID.
- This is transmitted back and forth between the user and the server via a session cookie.
- For a brand new session, PHP assigns an initially empty dictionary-style collection that can be used to hold any state values for this session.
- When the request processing is finished, the session state is saved to some type of state storage mechanism, called a session state provider
- When a new request is received for an already existing session, the session's dictionary collection is filled with the previously saved session data from the session state provider.

Session Storage

- It is possible to configure many aspects of sessions including where the session files are saved.
- The decision to save sessions to files rather than in memory (like ASP.NET) addresses the issue of memory usage that can occur on shared hosts as well as persistence between restarts.
- Inexpensive web hosts may sometimes stuff hundreds or even thousands of sites on each machine.
- Server memory may be storing not only session information, but pages being executed, and caching information.

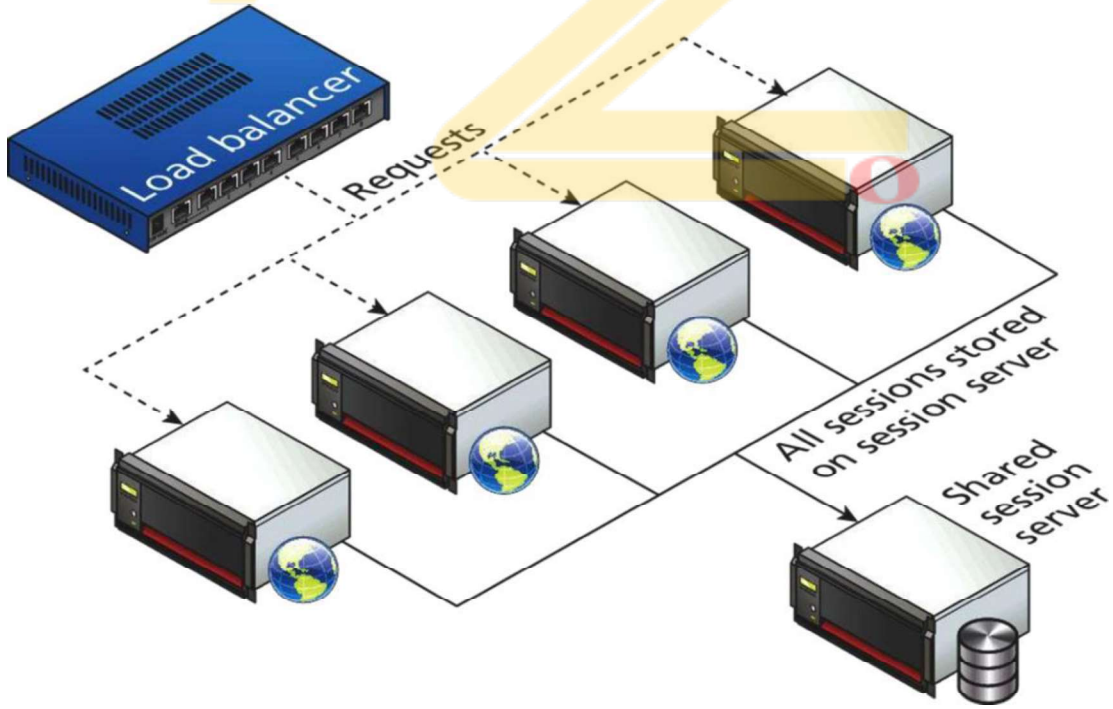


- Higher-volume web applications often run in an environment in which multiple web servers (also called a web farm) are servicing requests.
- In such a situation the in-process session state will not work, since one server may service one request for a particular session, and then a completely different server may service the next request for that session



There are effectively two categories of solution to this problem.

1. Configure the load balancer to be "session aware" and relate all requests using a session to the same server.
2. Use a shared location to store sessions, either in a database, mem cache, or some other shared session state mechanism



```
[Session]
; Handler used to store/retrieve data.
session.save_handler = memcache
session.save_path = "tcp://sessionServer:11211"
```

LISTING 13.7 Configuration in php.ini to use a shared location for sessions

HTML5 Web Storage

- Web storage is a new JavaScript only API introduced in HTML5.
- IT is meant to be a replacement to cookies, in that web storage is managed by the browser.
- Unlike cookies web storage data is not transported to and from the server with every request and response
- Web storage is not limited to 4k size barrier of cookies.
- Limit of 5 MB but browsers are allowed to store more per domain.
- Just as there were two types of cookies, there are two types of web storage
  - Local storage –It is for saving information that will persist between browser sessions.
  - Session storage – It is for information that will be lost once the browser session is finished.
- Below code illustrates the JavaScript code for writing information to web storage.
- There are two ways to store values in web storage
- Using setItem() function, or using the property shortcut(Eg: session storage.FavoriteArtist().

```
<form ... >
   <h1>Web Storage Writer</h1>
   <script language="javascript" type="text/javascript">

        if (typeof (localStorage) === "undefined" ||
              typeof (sessionStorage) === "undefined") {
            alert("Web Storage is not supported on this browser...");
        }
        else {
            sessionStorage.setItem("TodaysDate", new Date());
            sessionStorage.FavoriteArtist = "Matisse";

            localStorage.UserName = "Ricardo";
            document.write("web storage modified");
        }
   </script>
   <p><a href="WebStorageReader.php">Go to web storage reader</a></p>
</form>
```

LISTING 13.8 Writing web storage

- Below code illustrates the process of reading from web storage is equally straight forward.
- The difference between session Storage and local Storage in this example is that if you close the browser after writing and then run the code (Above code), only the local storage item will still contain a value.

```
<form id="form1" runat="server">
   <h1>Web Storage Reader</h1>
   <script language="javascript" type="text/javascript">

        if (typeof (localStorage) === "undefined" ||
              typeof (sessionStorage) === "undefined") {
            alert("Web Storage is not supported on this browser...");
        }
        else {
            var today = sessionStorage.getItem("TodaysDate");
            var artist = sessionStorage.FavoriteArtist;

            var user = localStorage.UserName;
            document.write("date saved=" + today);
            document.write("<br/>favorite artist=" + artist);
            document.write("<br/>user name = " + user);
        }
   </script>
</form>
```

LISTING 13.9 Reading web storage

- Cookies have the disadvantage of being limited in size, potentially disabled by the user, other security attacks and being sent in every single request and response to and from a given domain.
- Web storage is not as a cookie replacement but as a local cache for relatively static items available to JavaScript.
- One use of web storage is to store static content downloaded asynchronously such as xml or JSON from a web service in web storage, this reducing server load for subsequent requests by the session.
- Below fig illustrates an example of how web storage could be used as a mechanism for reducing server data request, there by speeding up the display of the page on the browser as well as reducing load on the server.

CACHING

- Caching is the vital way to improve the performance of web applications. Your browser uses caching to speed up the user experience by using locally stored versions of images and other files rather than re – requesting the files from the server.
- A server side developer only had limited control over browser caching.
- Why is this necessary?
- Every time a PHP page is requested, it must be fetched, parsed and executed by the PHP engine and end result is HTML that is sent back to the requestor.

- On way to address this problem is to cache the generated markup in server memory so that subsequent requests can be served from memory rather than from the execution of the page.
- There are two basic strategies to caching web applications.
  - PAGEOUTPUTCACHING

  It saves the rendered output of a page or user control and reuses the output instead of reprocessing the page when a user requests the page again
  - APPLICATIONDATACACHING

  It allows the developer to programmatically cache data.
- There are two models for page caching:
  - Full page caching
  - Partial page caching :

Only specific parts of page are cached while other parts are dynamically generated in the normal manner.

Application of data caching

- One of the biggest drawbacks with page output caching is that performance gains will only be had if the entire cached page is the same for numerous requests.
- An alternate strategy is to use application data caching in which a page will programmatically place commonly used collections of data that require time intensive queries from the database or web server into cache memory, and then other pages that also need that same data can use the cache version rather than re – retrieve it from its original location.
- While the default installation of PHP does not come with an application caching ability, a widely available free PECL extension called memcache is used for this ability.

Advanced JavaScript & JQuery

JAVASCRIPTPSEUDO-CLASSES

- JavaScript has no formal class mechanism, it does support objects (DOM).
- While most OO languages that support objects also support classes formally, JavaScript does not.
- Instead we define Pseudo–classes through a variety of syntax.

Using Object Literals

- An array in JavaScript can be instantiated var daysofweek = [“sun” , ”mon” ,“tue”…..];
- An object can be instantiated using object literals.

- **Object literals** are a list of key-value pairs with colons between the key and value with commas separating key-value pairs.
- Object literals are also known as **Plain Objects** in jQuery.
- JavaScript has no formal class mechanism.

Simple Variable var car ="Fiat";

Then this

var car = {type:"Fiat", model:500, color:"white"};

Properties car.name = Fiat car.model = 500 car.weight = 850kg car.color = white
Methods: car.start() car.drive() car.brake() car.stop()

Emulate Classes with functions

Use functions to encapsulate variables and methods together.

```
function Die(col) {
   this.color=col;
   this.faces=[1,2,3,4,5,6];
}
```

LISTING 15.1 Very simple Die pseudo-class definition as a function

Instantiation looks much like in PHP:

var oneDie = new Die("0000FF");

Adding methods to the objects

- To define a method in an object's function one can either define it internally or use a reference to a function define outside the class.
- One technique for adding a method inside of a class definition is by assigning an anonymous function to a variable

```
function Die(col) {
   this.color=col;
   this.faces=[1,2,3,4,5,6];

   // define method randomRoll as an anonymous function
   this.randomRoll = function() {
      var randNum = Math.floor((Math.random() * this.faces.length)+ 1);
      return faces[randNum-1];
   };
}
```

LISTING 15.2 Die pseudo-class with an internally defined method

Using Prototypes

So you can use a *prototype* of the class.

- Prototypes are used to make JavaScript behave more like an object-oriented language.
- The prototype properties and methods are defined *once* for all instances of an *object*.
- Every object has a prototype

```
// Start Die Class
function Die(col) {
   this.color=col;
   this.faces=[1,2,3,4,5,6];
}

Die.prototype.randomRoll = function() {
   var randNum = Math.floor((Math.random() * this.faces.length) + 1);
   return faces[randNum-1];
};
// End Die Class
```

LISTING 15.3 The Die pseudo-class using the prototype object to define methods

- This definition is better because it defines the method only once, no matter how many instance of die are created.
- How many ever created it will be reference to that one method. (fig below)

JQUERYFOUNDATIONS

- A **library** or **framework** is software that you can utilize in your own software, which provides some common implementations of standard ideas.
- Many developers find that once they start using a framework like jQuery, there"s no going back to "pure" JavaScript because the framework offers so many useful shortcuts and brief ways of doing things.

In August 2005 jQuery founder John Resig was looking into how to better combine **CSS selectors with succinct JavaScript notation**.

- Within 1 year AJAXand animations wereadded
- Additional modules
  - jQuery UI extension
  - mobile device support
- Continues to improve.

jQuery Selectors

- When discussing basic JavaScript we introduced the getElementByID () and querySelector() selector functions in JavaScript.
- Although the advanced querySelector() methods allow selection of DOM elements based on CSS selectors, it is only implemented in newest browsers
- jQuery introduces its own way to select an element, which under the hood supports a myriad of older browsers for you.

- ✓ The relationship between DOM objects and selectors is so important in JavaScript programming that the pseudo-class bearing the name of the framework, **jQuery()**
- ✓ Is reserved for selecting elements from the DOM.
- ✓ Because it is used so frequently, it has a shortcut notation and can be written as **$()**

Basic Selectors

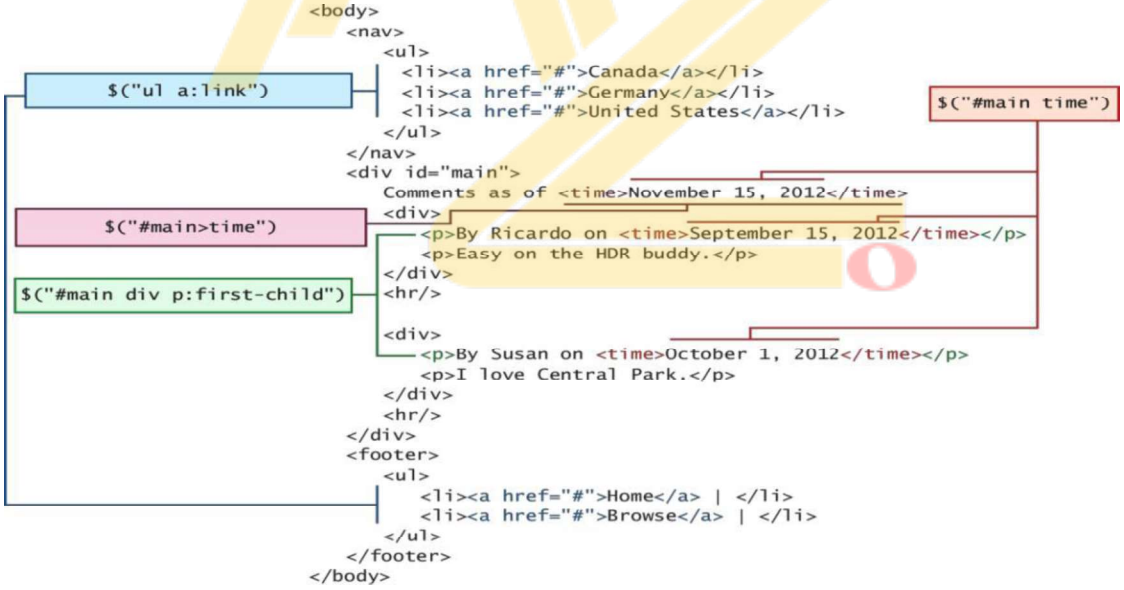
The four basic selectors are:

- $("*") Universal selector matches all elements (and is slow).
- $("tag") Element selector matches all elements with the given element name.
- $(".class") Class selector matches all elements with the given CSSclass.
- $("#id") Id selector matches all elements with a given HTML id attribute.
- For example, to select the single elementwith id="grab" you would write:
  var singleElement = $("#grab");
  Toget a set of all the <a> elements the selector would
  be: var allAs = $("a");
  These selectors replace the use of getElementById() entirely.

More CSS Selectors

In addition to these basic selectors, you can use the other CSSselectors:

- attribute selectors,
- pseudo-element selectors, and
- contextual selectors



Attribute Selector

Recall from CSS that you can select

- by attribute with square brackets
  - [attribute]
- Specify a value with an equals sign

    - [attribute=value]
- Search for a particular value in the beginning, end, or anywhere Inside a string
    - [attribute^=value]
    - [attribute$=value]
    - [attribute*=value]

Pseudo-Element Selector

Pseudo-element selectors are also from CSS and allow you to append to any selector using the colon and one of

- :link
- :visited
- :focus
- :hover
- :active
- :checked
- :first-child, :first-line, and :first-letter

Selecting all links that have been visited, for example, would be specified with:

var visitedLinks = $("a:visited");

Contextual Selector

Contextual selectors are also from CSS. Recall that these include:

- descendant (space)
- child (>)
- adjacent sibling (+)
- and general sibling (~).

To select all <p> elements inside of elements you would write

var para = $("div p");

HTML Properties

- Many HTML tags include *properties* as well as attributes, the most common being the *checked* property of a radio button or checkbox.
- The prop () method is the preferred way to retrieve and set the value of a property although, attr() may return some (less useful) values.
- <input **class="meh"** type="checkbox" checked="checked">

- Is accessed by jQuery as follows:
- **var theBox = $(".meh"); theBox.prop("checked");** // *evaluates to TRUE* **theBox.attr("checked")**; // *evaluates to "checked"*

Changing CSS

- jQuery provides the extremely intuitive **css()**methods.
- Toget a css value use the css() method with 1 parameter:

$color = $("#colourBox").**css**("**background-color**"); // ***get*** *thecolor*

- Toset a CSSvariable use css() with two parameters: the first being the CSSattribute, and the second the value.

// ***set*** *color to red*

$("#colourBox").**css**("**background-color**", "**#FF0000**");

jQuery Listeners

In JavaScript, you learned why having your **listeners** set up inside of the window.onload() event was a good practice. With jQuery we do the same thing but use the $(document).ready () event

```
$(document).ready(function(){
 //set up listeners on the change event for the file items.
 $("input[type=file]").change(function(){
     console.log("The file to upload is "+ this.value);
  });
});
```

LISTING 15.6 jQuery code to listen for file inputs changing, all inside the document's ready event

Modifying the DOM

- The append () method takes as a parameter an HTML string, a DOM object, or a jQuery object. That object is then added as the last child to the element(s) being selected.

HTML Before

```
<div class="external-links">
     <div class="linkOut">
        funwebdev.com
     </div>
     <div class="linkIn">
        /localpage.html
     </div>
     <div class="linkOut">
        pearson.com
     </div>
<div>
```

jQuery append

```
$(".linkOut").append(jsLink);
```

HTML After

```
<div class="external-links">
     <div class="linkOut">
        funwebdev.com
<a href='http://funwebdev.com'
title='jQuery'>Visit Us</a>
     </div>
     <div class="linkIn">
        /localpage.html
     </div>
     <div class="linkOut">
        pearson.com
<a href='http://funwebdev.com'
title='jQuery'>Visit Us</a>
     </div>
<div>
```

- A more advanced technique might make use of the content of each div being modified. In that case we use a cal back function in place of a simple element.
- The wrap () method is a cal back function, which is called for each element in a set (often an array).
- Each element then becomes this for the duration of one of the wrap () function's executions, allowing the unique title attributes as shown in Listing15.12.

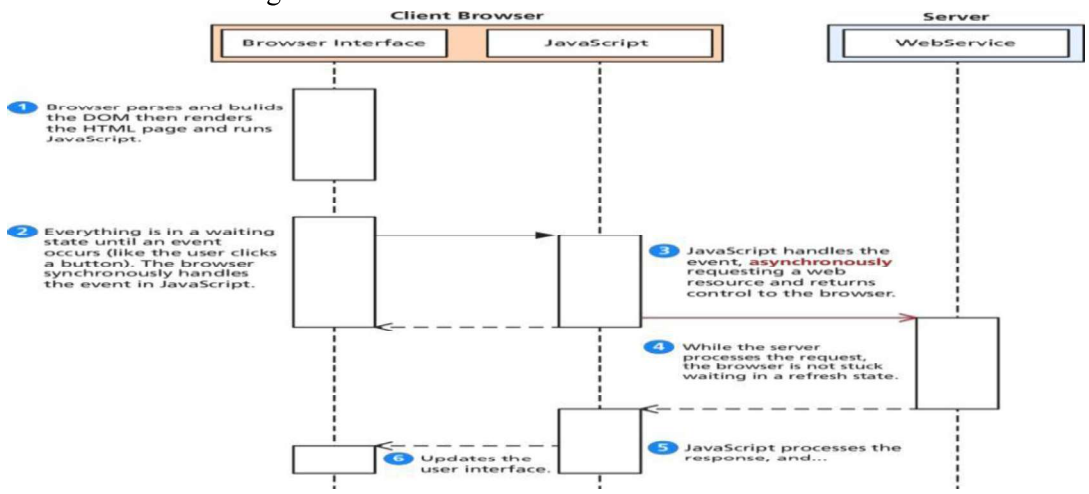
```
<div class="external-links">
 <div class="gallery">Uffuzi Museum</div>
 <div class="gallery">National Gallery</div>
 <div class="link-out">funwebdev.com</div>
</div>
```

```
<div class="external-links">
  <div class="galleryLink">
      <div class="gallery">Uffuzi Museum</div>
  </div>
   <div class="galleryLink">
      <div class="gallery">National Gallery</div>
  </div>
   <div class="link-out">funwebdev.com</div>
</div>
```

LISTING 15.10 HTML from Listing 15.9 modified by executing the wrap statement above

AJAX

- Asynchronous JavaScript with XML (AJAX) is a term used to describe a paradigm that allows a web browser to send messages back to the server without interrupting the flow of what's being shown in the browser.



- jQuery provides a family of methods to make asynchronous requests. We will start simple and work our way up.
- Consider the very simple server time example we just saw. If currentTime.php returns a single string and you want to load that value asynchronously

jQuery.get (**url [, data] [, success(data, textStatus, jqXHR)] [, dataType]**) **url** is a string that holds the location to send the request. **data** is an optional parameter that is a query string or a *PlainObject.* **success(data,textStatus,jqXHR)** is an optional *callback* function that executes when the response is received. **data** holding the body of the response as astring. **textStatus** holding the status of the request (i.e.,"success"). **jqXHR** holding a jqXHR object, described shortly. **dataType** is an optional parameter to hold the type of data expected from the server.

```
$.get("/vote.php?option=C", function(data,textStatus,jsXHR) {
  if (textStatus=="success") {
     console.log("success! response is:" + data);
  }
  else {
     console.log("There was an error code"+jsXHR.status);
  }
  console.log("all done");
});
```

LISTING 15.13 jQuery to asynchronously get a URL and outputs when the response arrives

All of the $.get() requests made by jQuery return a **jqXHR** object to encapsulate the response from theserver. In practice that means the data being referred to in the cal back from Listing 15.13 is actually an object with backward compatibility with XMLHttpRequest

- **Abort ()** stops execution and prevents any call-back or handlers from receiving the trigger to execute.
- **getResponseHeader()** takes a parameter and gets the current value of that header.
- **readyState** is an integer from 1 to 4 representing the state of the request. The values include 1: successful call open() 2: sending, 3: response being processed, and 4: completed.
- **responseXML**and/or **responseText** the main response to the request.
- **setRequestHeader(name, value)** when used before actually instantiating the request allows headers to be changed for the request.
- **status** is the HTTPrequest status codes (200 =ok)

- **statusText** is the associated description of the statuscode.

POST requests

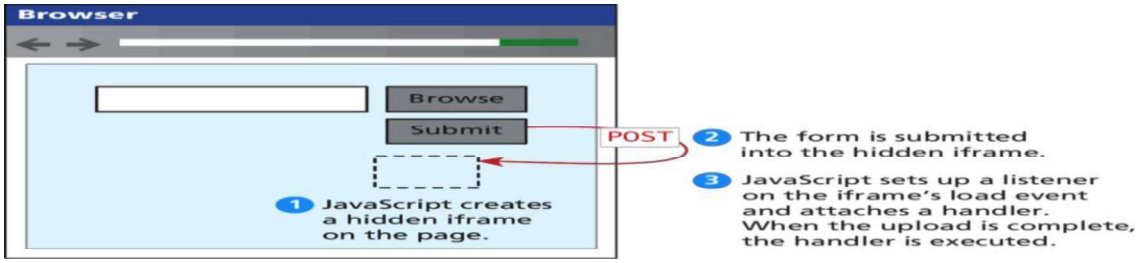
- POST requests are often preferred to GET requests because one can post an unlimited amount of data, and because they do not generate viewable URLs for each action.
- GET requests are typically not used when we have forms because of the messy URLs and that limitation on how much data we can transmit.
- With POST it is possible to transmit files, something which is not possible with GET.
- The HTTP 1.1 definition describes GET as a **safe method meaning** that they should not change anything, and should only read data.
- POSTs on the other hand are not safe, and should be used whenever we are changing the state of our system (like casting avote). get() method.
- POST syntax is almost identical to GET.
- jQuery.**post** (url [, data] [, success(data, textStatus, jqXHR)] [, dataType])
- If we were to convert our vote casting code it would simply change the first line from
- var jqxhr = $.get("/vote.php?option=C"); to
  var jqxhr = $.post("/vote.php", "option=C");

CORS

- Since modern browsers prevent cross-origin requests by default (which is good for security), sharing content legitimately between two domains becomes harder.
- Cross-origin resource sharing (CORS) uses new headers in the HTML5 standard implemented in most new browsers.
- If a site wants to allow any domain to access its content through JavaScript, it would add the following header to all of its responses.
- Access-Control-Allow-Origin:
- A better usage is to specify specific domains that are allowed, rather than cast the gates open to each and every domain. To allow our domain to make cross site requests we would add the header:
- Access-Control-Allow-Origin: www.funwebdev.com
- Rather than the wildcard*.

ASYNCHRONOUSFILE TRANSMISSION

The original workaround to allow the asynchronous posting of files was to use a hidden <iframe> element to receive the posted files.



```
<form name="fileUpload" id="fileUpload" enctype="multipart/form-data"
      method="post" action="upload.php">
<input name="images" id="images" type="file" multiple />
<input type="submit" name="submit" value="Upload files!"/>
</form>
```

LISTING 15.17 Simple file upload form

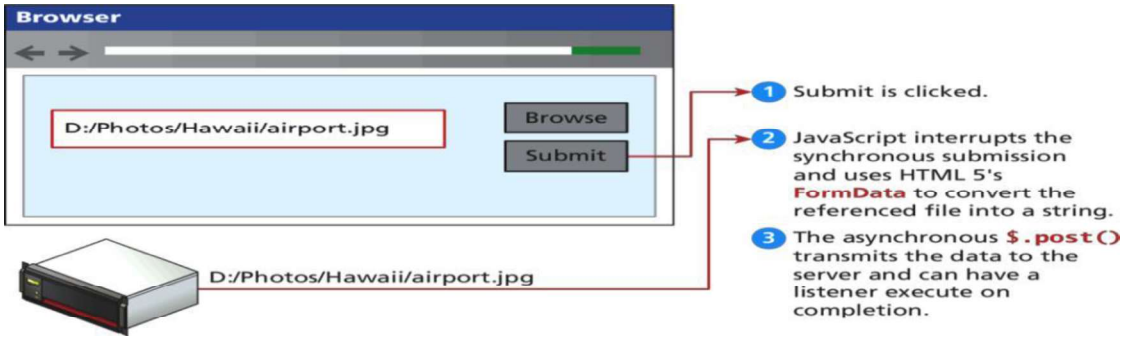
```
$(document).ready(function() {
  // set up listener when the file changes
  $(":file").on("change",uploadFile);
  // hide the submit buttons
  $("input[type=submit]").css("display","none");
});

// function called when the file being chosen changes
function uploadFile () {
  // create a hidden iframe
  var hidName = "hiddenIFrame";
  $("#fileUpload").append("<iframe id='"+hidName+"' name='"+hidName+"'
  style='display:none' src='#' ></iframe>");

  // set form's target to iframe
  $("#fileUpload").prop("target",hidName);
  // submit the form, now that an image is in it.
  $("#fileUpload").submit();
  // Now register the load event of the iframe to give feedback
  $('#'+hidName).load(function() {
  var link = $(this).contents().find('body')[0].innerHTML;
  // add an image dynamically to the page from the file just uploaded
  $("#fileUpload").append("<img src='"+link+"' />");
  });
}
```

LISTING 15.18 Hidden iFrame technique to upload files

Using the **FormData** interface and File API, which is part of HTML5, you no longer have to trick the browser into posting your file data asynchronously.



- When we consider uploading multiple files, you may want to upload a single file, rather than the entire form every time. To support that pattern, you can access a single file and post it by appending the raw file to a FormData object.
- The advantage of this technique is that you submit each file to the server asynchronously as the user changes it; and it allows multiple files to be transmitted at once.

```
var xhr = new XMLHttpRequest();
// reference to the 1st file input field
var theFile = $(":file")[0].files[0];
var formData = new FormData();
formData.append('images', theFile);
```

LISTING 15.20 Posting a single file from a form

```
var allFiles =  $(":file")[0].files;
for (var i=0;i<allFiles.length;i++) {
  formData.append('images[]', allFiles[i]);
}
```

LISTING 15.21 Looping through multiple files in a file input and appending the data for posting

ANIMATION

The hide () and show () methods can be called with no arguments to perform a default animation. Another version allows two parameters: the duration of the animation (in milliseconds) and a cal back method to execute on completion.

The fadeIn() and fadeOut() shortcut methods control the opacity of an element. The parameters passed are the duration and the cal back, just like hide () and show (). Unlike hide () and show (), there is no scaling of the element, just strictly control over the transparency.

As you may have seen, the shortcut methods come in pairs, which make them ideal for toggling between a shown and hidden state. Using a toggle method means you don't have to check the current state and then conditionally call one of the two methods;

- To toggle between the visible and hidden states you can use the **toggle ()** methods.
- To toggle between fading in and fading out, use the **fade Toggle()** method
- To toggle between the two sliding states can be achieved using the **slideToggle()**method.

Raw Animation

- The animate() method has several versions, but the one we will look at has the following form:
- .animate(properties, options);
- The properties parameter contains a Plain Object with all the CSSstyles of the final state of the animation.
- The options parameter contains another Plain Object with any of the following options set:
- Always is the function to be called when the animation completes or stops with a fail condition. This function will always be called (hence the name).
- Done is a function to be called when the animation completes.
- Duration is a number controlling the duration of the animation.
- Fail is the function called if the animation does not complete.
- Progress is a function to be called after each step of the animation.

- Queue is a Boolean value telling the animation whether to wait in the queue of animations or not. If false, the animation begins immediately.
- Step is a function you can define that will be called periodically while the animation is still going.
- Advanced options called easing and special Easing allow for advanced control over the speed of animation.

In web development, easing functions are used to simulate that natural type of movement. They are mathematical equations that describe how fast or slow the transitions occur at various points during the animation.

Included in jQuery are

- linear
- swing
- Easing functions.

BACKBONEMVC FRAMEWORKS

- Backbone is an MVC framework that further abstracts JavaScript with libraries intended to adhere more closely to the MVC model.

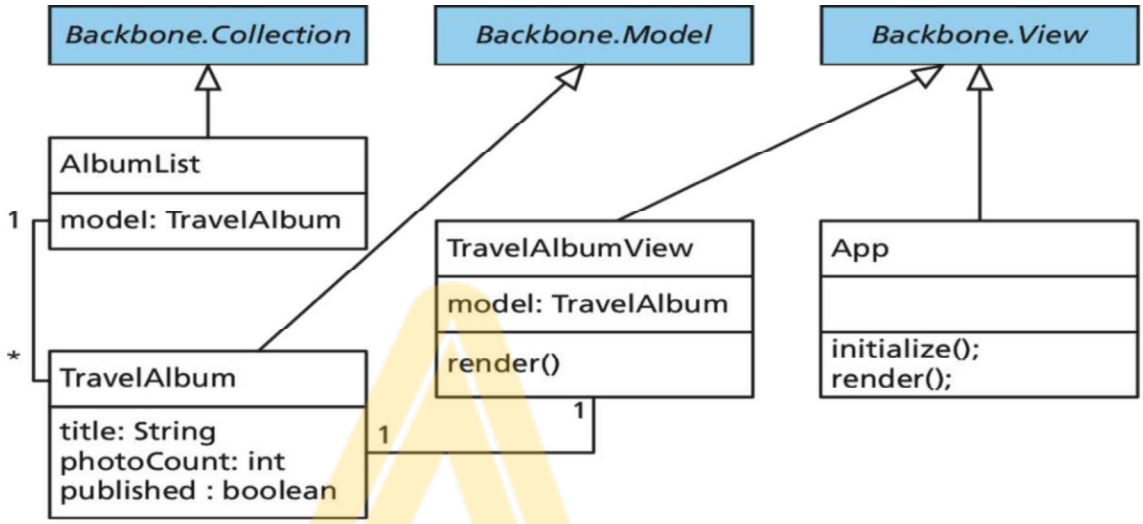
  Include with:

  <script src="underscore-min.js"></script>

  <script src="backbone-min.js"></script>

- In Backbone, you build your client scripts around the concept of **models**.
- Backbone.js defines **models** as *the heart of any JavaScript application, containing the interactive data as well as a large part of the logic surrounding it: conversions, validations, computed properties, and access control.*
- The Models you define using Backbone must *extend* Backbone Model

- ✓ In addition to models, Backbone introduces the concept of **Collections**, which are normally used to contain lists of Model objects.
- ✓ These collections have advanced features and like a database can have indexes to improve search performance.
- ✓ A collection is defined by extending from Backbone's Collection object.



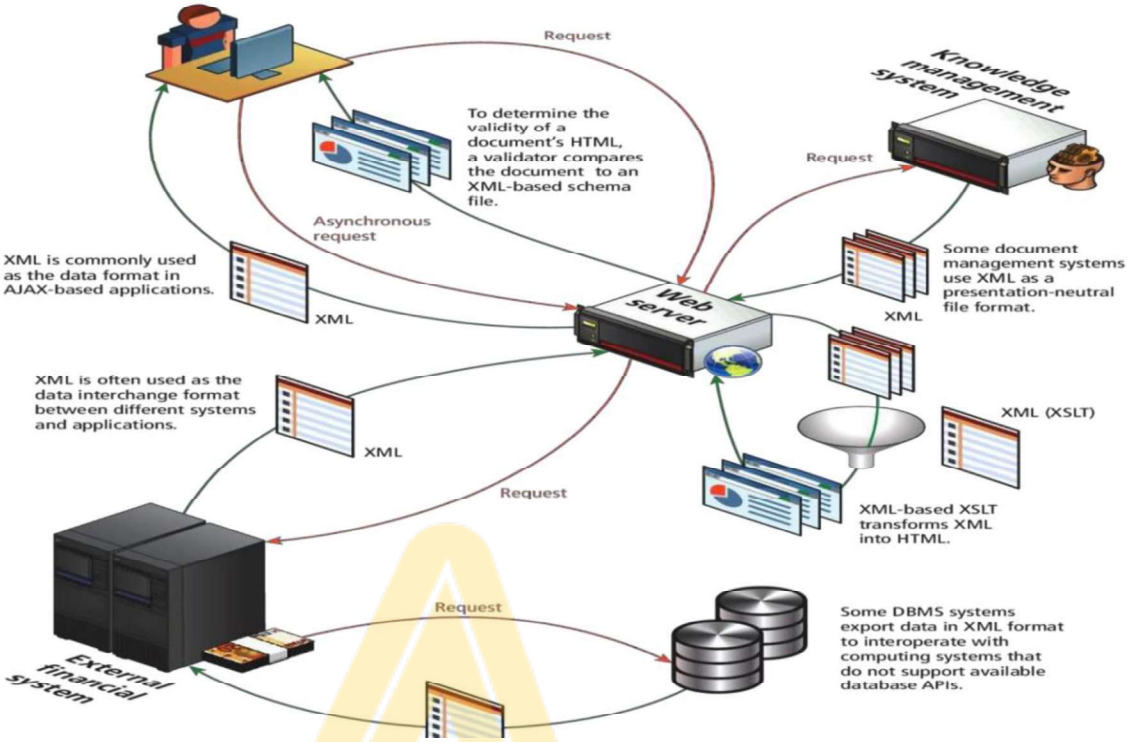

```
// Create a model for the albums
var TravelAlbum = Backbone.Model.extend({
   defaults:{
            title: 'NewAlbum',
            photoCount: 0,
            published: false
        },

        // Function to publish/unpublish
        toggle: function(){
            this.set('checked', !this.get('checked'));
        }
    });
```

LISTING 15.25 A PhotoAlbum Model extending from Backbone.Model

XML Processing and Web Services

- ❑ XML is a markup language, but unlike HTML, XML can be used to mark up any type of data.
- ❑ Benefits of XML data is that as plain text, it can be read and transferred between applications and different operating systems as well as being human-readable.
- ❑ XML is used on the web server to communicate asynchronously with the browser
- ❑ Used as a data interchange format for moving information between systems



Well Formed XML

For a document to be **well-formed XML**, it must follow the syntax rules for XML:

- Element names are composed of any of the valid characters (most punctuation symbols and spaces are not allowed) in XML.
- Element names can't start with number.
- There must be a single-root element. A **root element** is one that contains all the other elements; for instance, in an HTML document, the root element is<html>.
- All elements must have a closing element (or be self-closing).
- Elements must be properly nested.
- Elements can contain attributes.
- Attribute values must always be within quotes.
- Element and attribute names are case sensitive.
- A **valid XML** document is one that is well formed and whose element and content conform to a document type definition (DTD) or its schema.
- A DTD tells the XML parser which elements and attributes to expect in the document as well as the order and nesting of those elements.
- A DTD can be defined within an XML document or within an external file.

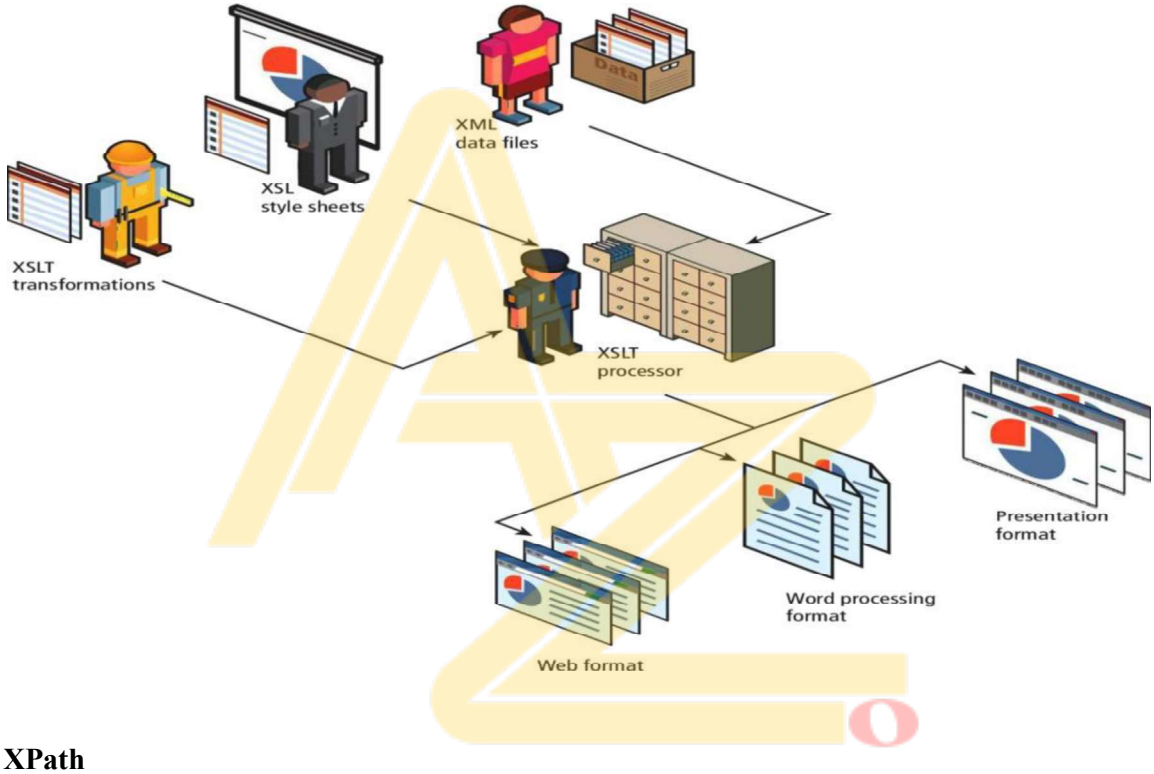
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE art [
<!ELEMENT art (painting*)>
<!ELEMENT painting (title,artist,year,medium)>
<!ATTLIST painting id CDATA #REQUIRED>
<!ELEMENT title (#PCDATA)>
<!ELEMENT artist (name,nationality)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT nationality (#PCDATA)>
<!ELEMENT year (#PCDATA)>
<!ELEMENT medium (#PCDATA)>
]>
<art>
...
</art>
```

LISTING 17.2 Example DTD

- The main drawback with DTD is that they can only validate the existence and ordering of elements. They provide no way to validate the values of attributes or the textual content of elements.
- For this type of validation, one must instead use XML schemas, which have the added advantage of using XML syntax. Unfortunately, schemas have the corresponding disadvantage of being long-winded and harder for humans to read and comprehend; for this reason, they are typically created with tools.

XSLT

XSLT is an XML-based programming language that is used for transforming XML into other document formats



XPath

- **XPath** is a standardized syntax for searching an XML document and for navigating to elements within the XML document
- XPath is typically used as part of the programmatic manipulation of an XML document in PHP and other languages
- XPath uses a syntax that is similar to the one used in most operating systems to access directories.

XML PROCESSING

XML processing in PHP, JavaScript and other modern development environments is divided into two basic styles:

- The **in-memory approach**, which involves reading the entire XML file into memory into some type of data structure with functions for accessing and manipulating the data.
- The **event or pull approach**, which lets you pull in just a few elements or lines at a time, thereby avoiding the memory load of large XML files.
- All modern browsers have a built-in XML parser and their JavaScript implementations support an in-memory XML DOM API.

You can use the already familiar DOM functions such as

- getElementById(),
- getElementsByTagName()
- createElement()

To access and manipulate the data.

```
<script>
if (window.XMLHttpRequest)  {
  // code for IE7+, Firefox, Chrome, Opera, Safari
  xmlhttp=new XMLHttpRequest();
}
else  {
  // code for old versions of IE (optional you might just decide to
  // ignore these)
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
}
// load the external XML file
xmlhttp.open("GET","art.xml",false);
xmlhttp.send();
xmlDoc=xmlhttp.responseXML;

// now extract a node list of all <painting> elements
paintings = xmlDoc.getElementsByTagName("painting");
if (paintings) {
    // loop through each painting element
    for (var i = 0; i < paintings.length; i++)
    {
        // display its id attribute
        alert("id="+paintings[i].getAttribute("id"));

        // find its <title> element
        title = paintings[i].getElementsByTagName("title");
        if (title) {
            // display the text content of the <title> element
            alert("title="+title[0].textContent);
        }
    }
}
</script>
```

LISTING 17.5 Loading and processing an XML document via JavaScript

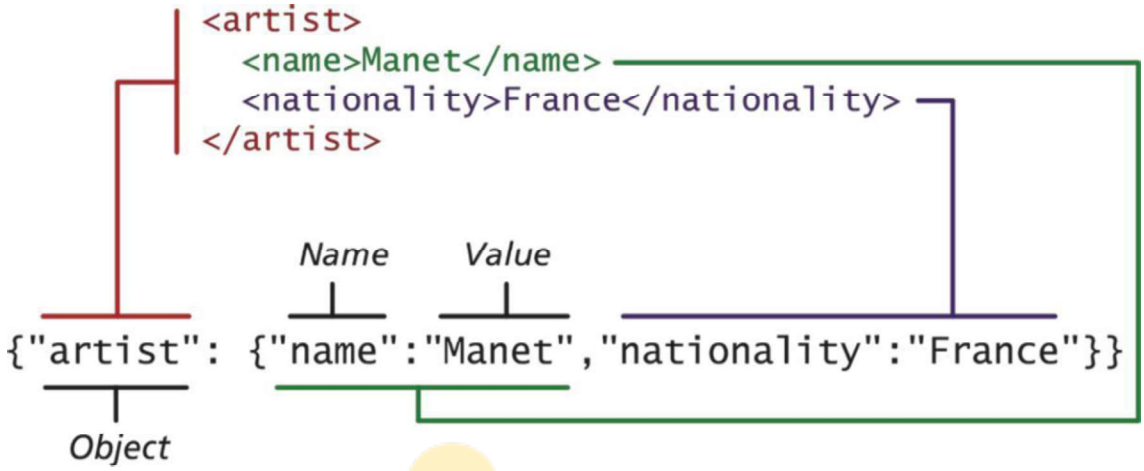
PHP provides several extensions or APIs for working with XML including:

- The **SimpleXML** extension which loads the data into an object that allows the developer to access the data via array properties and modifying the data via methods.
- The **XMLReader** is a read-only pull-type extension that uses a cursor-like approach similar to that used with database processing

JSON

- **JSON** stands for JavaScript Object Notation (though its use is not limited to JavaScript)
- Like XML, JSON is a data serialization format. It provides a more concise format than XML.

- Many REST web services encode their returned data in the JSON data format instead of XML.



OVERVIEW OF WEBSERVICES

- Web services are the most common example of a computing paradigm commonly referred to as service- oriented computing (SOC).
- A service is a piece of software with a platform- independent interface that can be dynamically located and invoked.
- Web services are a relatively standardized mechanism by which one software application can connect to and communicate with another software application using web protocols.
- They can provide interoperability between different software applications running on different platforms
- They can be used to implement service-oriented architecture (SOA)
- They can be offered by different systems within an organization as well as by different organizations

SOAP is the message protocol used to encode the service invocations and their return values via XML within the HTTP header.

- SOAP and WSDL are complex XML schemas
- akin to using a compiler: its output may be complicated to understand
- The enthusiasm for SOAP-based web services had

cooled. REST stands for Representational State Transfer.

- REST full web service does away with the service description layer, and needs no separate protocol for encoding message requests and responses.
- It simply uses HTTP URLs for requesting a resource/object (and for encoding input parameters).
- The serialized representation of this object, usually an XML or JSON stream, is then returned to the requestor as a normal HTTP response.

- REST appears to have almost completely displaced SOAP services.

Identifying and Authenticating Service Requests

Most web services are not open. Instead they typically employ one of the following techniques:

- **Identity**. Each web service request must identify who is making the request. **Authentication**. Each web service request must provide additional evidence that they are who they say they are
- Some web services are providing private/proprietary information or are involving financial transactions.
- In this case, these services not only may require an API key, but they also require some type of user name and password in order to perform an authorization.
- Many of the most well-known web services instead make use of the O Auth standard.