Posted in: [Using jQuery Core](http://learn.jquery.com/using-jquery-core/)

**$( document ).ready()**

A page can't be manipulated safely until the document is "ready." jQuery detects this state of readiness for you. Code included inside $( document ).ready() will only run once the page Document Object Model (DOM) is ready for JavaScript code to execute. Code included inside $( window ).load(function() { ... }) will run once the entire page (images or iframes), not just the DOM, is ready.

|  |  |
| --- | --- |
| 1  2  3  4 | *// A $( document ).ready() block.*  $( document ).ready(**function**() {  console.log( "ready!" );  }); |

A function to execute after the DOM is ready. - [jquery.com](http://jquery.com/)  
  
You should wrap all your javascript code with this function to ensure that the code only runs when the page is fully rendered.   
  
Without it you could run into errors where the JavaScript can't find elements as they haven't rendered to the page yet.   
  
Not the same as window.load which is required when all the pages assets like images are fully rendered as well as the DOM.   
  
Also place your scripts at the bottom of your page for faster page load.

JSON: **J**ava**S**cript **O**bject **N**otation.

JSON is a syntax for storing and exchanging data.

JSON is an easier-to-use alternative to XML.

AJAX = Asynchronous JavaScript and XML.

AJAX is not a new programming language, but a new way to use existing standards.

AJAX is the art of exchanging data with a server, and updating parts of a web page - without reloading the whole page.

The var\_dump() function is used to display structured information (type and value) about one or more variables.

## Syntax

var\_dump(variable1, variabl2, ....variablen)

## **Decoding JSON in PHP (json\_decode)**

PHP json\_decode() function is used for decoding JSON in PHP. This function returns the value decoded from json to appropriate PHP type.

### **SYNTAX:**

mixed json\_decode ($json [,$assoc = false [, $depth = 512 [, $options = 0 ]]])

### **PARAMATERS:**

* **json\_string**: It is encoded string which must be UTF-8 encoded data
* **assoc**: It is a boolean type parameter, when set to TRUE, returned objects will be converted into associative arrays.
* **depth**: It is an integer type parameter which specifies recursion depth
* **options**: It is an integer type bitmask of JSON decode, JSON\_BIGINT\_AS\_STRING is supported.

### **EXAMPLE**

The following example shows how PHP can be used to decode JSON objects:

<?php

$json = '{"a":1,"b":2,"c":3,"d":4,"e":5}';

var\_dump(json\_decode($json));

var\_dump(json\_decode($json, true));

?>

While executing, this will produce following result:

object(stdClass)#1 (5) {

["a"] => int(1)

["b"] => int(2)

["c"] => int(3)

["d"] => int(4)

["e"] => int(5)

}

array(5) {

["a"] => int(1)

["b"] => int(2)

["c"] => int(3)

["d"] => int(4)

["e"] => int(5)

}

<tr> This is a table row

class Leo

{

public function \_\_construct()

{

}

public function hello()

{

echo "Hello Leo";

}

private function priv()

{

echo "I am private";

}

protected function prot()

{

echo "I am protected";

}

}

class LeoPet extends Leo {

public $name = "Messi";

public function \_\_construct()

{

$this->hello();

$this->prot();

}

}

/\*$obj = new Leo();

$obj->hello();

//$obj->priv();

$obj->prot();\*/

$pet = new LeoPet();

$pet->hello();

$pet->prot();

var\_dump($pet);

exit;

The visibility of a property or method can be defined by prefixing the declaration with the keywords public,protected or private.

Class members declared public can be accessed everywhere.

Members declared protected can be accessed only within the class itself and by inherited and parent classes.

Members declared as private may only be accessed by the class that defines the member.

Structure[[edit](http://en.wikipedia.org/w/index.php?title=Query_string&action=edit&section=1" \o "Edit section: Structure)]

A typical URL containing a query string is as follows:

http://example.com/over/there?name=ferret

When a server receives a request for such a page, it may run a program, passing the query\_string unchanged to the program. The first question mark is used as a separator and is not part of the query string.[[2]](http://en.wikipedia.org/wiki/Query_string#cite_note-2)[[3]](http://en.wikipedia.org/wiki/Query_string#cite_note-3)

SFTP

Use Filezilla or WinSCP

Use var\_dump(); /\* use this to get the data out \*/

SSL (Secure Sockets Layer) Certificate

SSL stands for Secure Sockets Layer. It provides a secure connection between internet browsers and websites, allowing you to transmit private data online. Sites secured with SSL display a padlock in the browsers URL and possibly a green address bar if secured by an EV Certificate.

what is SSL used for? The SSL protocol is used by millions of e-Business providers to protect their customers, ensuring their online transactions remain confidential. A web page should use encryption expected to submit confidential data, including credit card details, passwords or any personal information. All web browsers have the ability to interact with secured sites so long as the site’s certificate is from a recognized certificate authority, such as Comodo.

In the context of an [HTTP](http://en.wikipedia.org/wiki/HTTP) transaction, **basic access authentication** is a method for an [HTTP user agent](http://en.wikipedia.org/wiki/HTTP_user_agent) to provide a [user name](http://en.wikipedia.org/wiki/User_name) and [password](http://en.wikipedia.org/wiki/Password" \o "Password)when making a request.

**HTTP header fields** are components of the header section of [request](http://en.wikipedia.org/wiki/HTTP_request) and response messages in the [Hypertext Transfer Protocol](http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol) (HTTP). They define the operating parameters of an HTTP transaction.

OAuth 2.0 defines a set of endpoints. An endpoint is typically a URI on a web server. For instance, the address of a Java servlet, JSP page, PHP page, ASP.NET page etc.

The endpoints defined are:

* Authorization Endpoint
* Token Endpoint
* Redirection Endpoint
* The authorization endpoint and token endpoint are both located on the authorization server. The redirection endpoint is located in the client application. Each of these endpoints are covered below.
* The endpoints are illustrated in this diagram:

|  |
| --- |
| OAuth 2.0 Endpoints. |
| **OAuth 2.0 Endpoints.** |

* The OAuth 2.0 specification does not describe how the URI of these endpoints are found or documented. That is up to each implementer to decide. Most sites will have a subsite for developers documenting these endpoints.

## Authorization Endpoint

The authorization endpoint is the endpoint on the authorization server where the resource owner logs in, and grants authorization to the client application.

## Token Endpoint

The token endpoint is the endpoint on the authorization server where the client application exchanges the authorization code, client ID and client secret, for an access token.

## Redirect Endpoint

The redirect endpoint is the endpoint in the client application where the resource owner is redirected to, after having granted authorization at the authorization endpoint.

Short for ***H***yper***T***ext***T***ransfer***P***rotocol, HTTP is the underlying[protocol](http://www.webopedia.com/TERM/P/protocol.html) used by the [World Wide Web](http://www.webopedia.com/TERM/W/World_Wide_Web.html). HTTP defines how messages are formatted and transmitted, and what actions [Web servers](http://www.webopedia.com/TERM/W/Web_server.html) and [browsers](http://www.webopedia.com/TERM/B/browser.html) should take in response to various commands. For example, when you enter a [URL](http://www.webopedia.com/TERM/U/URL.html) in your browser, this actually sends an HTTP command to the Web server directing it to fetch and transmit the requested [Web page](http://www.webopedia.com/TERM/W/web_page.html).

## *HTTP Status Codes*

Errors on the Internet can be quite frustrating — especially if you do not know the difference between a 404 error and a 502 error. These error messages, also called HTTP status codes are response codes given by Web servers and help identify the cause of the problem.

For example, "404 File Not Found" is a common HTTP status code. It means the Web server cannot find the file you requested. The file -- the webpage or other document you try to load in your Web browser --  has either been moved or deleted, or you entered the wrong URL or document name.

Knowing the meaning of the HTTP status code can help you figure out what went wrong. On a 404 error, for example, you could look at the URL to see if a word looks misspelled, then correct it and try it again. If that doesn't work backtrack by deleting information between each backslash, until you come to a page on that site that isn't a 404. From there you may be able to find the page you're looking for.

# PHP Header Function

When you request a web page be brought back to your browser, you're not just bringing back the web page. You're also bringing back something called a HTTP HEADER. This is some extra information, such as type of programme making the request, date requested, should it be displayed as a HTML document, how long the document is, and a lot more besides.

One of the things HTTP HEADER also does is to give status information. This could be whether the page was found (404 errors), and the location of the document. If you want to redirect your users to another page, here's an example:

**<?PHP**

**header("Location: http://www.homeandlearn.co.uk/");**

**?>**

**<html>**

**<body>  
</body>**

**</html>**

Note how the header code goes before any HTML. If you put header code after the HTML, you'll get an error along the lines of "Cannot modify header information."

## The POST Method

The POST method transfers information via HTTP headers. The information is encoded as described in case of GET method and put into a header called QUERY\_STRING.

* The POST method does not have any restriction on data size to be sent.
* The POST method can be used to send ASCII as well as binary data.
* The data sent by POST method goes through HTTP header so security depends on HTTP protocol. By using Secure HTTP you can make sure that your information is secure.
* The PHP provides **$\_POST** associative array to access all the sent information using POST method.

URL encoding converts characters into a format that can be transmitted over the Internet.

In order to use the object operator, you will need to create and instantiate a class, as follows:

class MyClass {

public $myVar;

public function myMethod() {

}

}

$instance = new MyClass();

$instance->myVar = "Hello World"; // Assign "Hello World" to "myVar"

$instance->myMethod(); // Run "myMethod()"

Let me explain the above code:

1. First, a class with a name of "MyClass" is created, with a variable of "myVar" and a method (basically a function within a class) with a name of "myMethod".
2. "$instance" is created, and then it is assigned a new instance of the "MyClass" class.
3. $instance->myVar, with the object operator accesses the public instance variable within the $instance object, and assigns it a value of "Hello World". Similarly, the "myMethod" is called within the $instance object, also using the object operator.

The object operator is simply PHPs way of accessing, running, or assigning "stuff" within an object.

Creating a job application:

PHP: Hypertext PreProcessor

[www.tiobe.com](http://www.tiobe.com) – ranking of programming languages

<http://windows.php.net/> (windows binaries)

<https://www.apachefriends.org/index.html>

XAMPP Apache + MYSQL + PHP + Perl

PREG – pearl regular expression

Include\_once – only the first time it will load and execute

Require\_once – checks once the file

State Management

* Mechanism to store data across requests
* Use cookies with a unique ID to identify/remember users

// start session support

Session\_start();

// write to session

$\_SESSION[‘key’] = ‘value’;

The password hasing api makes storing one-way encrypted passwords easy

Sessions are used to store data on the server between requests

The **htmlspecialchars** function in PHP is used to convert 5 characters into corresponding HTML entities where applicable. It is used to encode user input on a website so that users cannot insert harmful HTML codes into a site

[http://beta3.a8websites.co.uk/luggage/samsonite-firelite-spinner-55cm-20.html#](http://beta3.a8websites.co.uk/luggage/samsonite-firelite-spinner-55cm-20.html)

[beta16/leo/a8websit@beta3.a8websites.co.uk](mailto:beta16/leo/a8websit@beta3.a8websites.co.uk)

custom.js

**isset** – Determine if a variable is set and is not null

 $\_POST are used to collect form-data using the HTTP POST method

## **GET vs. POST**

Both GET and POST create an array (e.g. array( key => value, key2 => value2, key3 => value3, ...)). This array holds key/value pairs, where keys are the names of the form controls and values are the input data from the user.

Both GET and POST are treated as $\_GET and $\_POST. These are superglobals, which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.

$\_GET is an array of variables passed to the current script via the URL parameters.

$\_POST is an array of variables passed to the current script via the HTTP POST method.

## **When to use GET?**

Information sent from a form with the GET method is **visible to everyone** (all variable names and values are displayed in the URL). GET also has limits on the amount of information to send. The limitation is about 2000 characters. However, because the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases.

GET may be used for sending non-sensitive data.

**Note:** GET should NEVER be used for sending passwords or other sensitive information!

## **When to use POST?**

Information sent from a form with the POST method is **invisible to others** (all names/values are embedded within the body of the HTTP request) and has **no limits** on the amount of information to send.

Moreover POST supports advanced functionality such as support for multi-part binary input while uploading files to server.

However, because the variables are not displayed in the URL, it is not possible to bookmark the page.

preg\_match — Perform a regular expression match

filter\_var — Filters a variable with a specified filter

## **Error Messages Explained**

PHP returns an appropriate error code along with the file array. The error code can be found in the error segment of the file array that is created during the file upload by PHP. In other words, the error might be found in[$\_FILES['userfile']['error']](http://php.net/manual/en/reserved.variables.files.php).

## **Definition and Usage**

The mysqli\_connect\_errno() function returns the error code from the last connection error, if any.

## **Syntax**

mysqli\_connect\_errno();

[dblink](http://www.postgresql.org/docs/9.1/static/contrib-dblink-function.html) -- executes a query in a remote database

The function adds an escape character, the backslash, \, before certain potentially dangerous characters in a string passed in to the function. The characters escaped are

\x00, \n, \r, \, ', " and \x1a.

This can help prevent SQL injection attacks which are often performed by using the ' character to append malicious code to an SQL query.

file\_get\_contents — Reads entire file into a string

intval — Get the integer value of a variable

htmlspecialchars — Convert special characters to HTML entities

## **Use htmlspecialchars**

PHP has a function called [htmlspecialchars()](http://php.net/manual/en/function.htmlspecialchars.php" \t "_blank) which converts certain characters into HTML entities.  Special characters like

*<*

are converted to their HTML entities such as

*&lt;*

HTML entities are safe to output to the page, because it displays the characters as literals so that browsers do not interpret them as HTML tags.

The include statement includes and evaluates the specified file.

session\_start — Start new or resume existing session

A session is a way to store information (in variables) to be used across multiple pages.

Unlike a cookie, the information is not stored on the users computer.

## **What is a PHP Session?**

When you work with an application, you open it, do some changes, and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are or what you do, because the HTTP address doesn't maintain state.

Session variables solve this problem by storing user information to be used across multiple pages (e.g. username, favorite color, etc). By default, session variables last until the user closes the browser.

So; Session variables hold information about one single user, and are available to all pages in one application.

## **Start a PHP Session**

A session is started with the session\_start() function.

Session variables are set with the PHP global variable: $\_SESSION.

Now, let's create a new page called "demo\_session1.php". In this page, we start a new PHP session and set some session variables:

## **Example**

<?php  
// Start the session  
session\_start();  
?>  
<!DOCTYPE html>  
<html>  
<body>  
  
<?php  
// Set session variables  
$\_SESSION["favcolor"] = "green";  
$\_SESSION["favanimal"] = "cat";  
echo "Session variables are set.";  
?>  
  
</body>  
</html>

**How does it work? How does it know it's me?**  
  
Most sessions set a user-key on the user's computer that looks something like this: 765487cf34ert8dede5a562e4f3a7e12. Then, when a session is opened on another page, it scans the computer for a user-key. If there is a match, it accesses that session, if not, it starts a new session.

## **Destroy a PHP Session**

To remove all global session variables and destroy the session, use session\_unset() and session\_destroy():

## **Example**

<?php  
session\_start();  
?>  
<!DOCTYPE html>  
<html>  
<body>  
  
<?php  
// remove all session variables  
session\_unset();   
  
// destroy the session   
session\_destroy();   
?>  
  
</body>  
</html>

$\_SERVER is an array which holds information of headers, paths, script locations. Web server creates the entries in the array. This is not assured that every web server will provide similar information, rather some servers may include or exclude some information which are not listed here.

$\_SERVER has following basic properties:

1. Set by web server.

2. Directly related to the runtime environment of the current php script.

3. It does the same job as $HTTP\_SERVER\_VARS used to do in previous versions of PHP

## **PHP Global Variables - Superglobals**

Several predefined variables in PHP are "superglobals", which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.

The PHP superglobal variables are:

* $GLOBALS
* $\_SERVER
* $\_REQUEST
* $\_POST
* $\_GET
* $\_FILES
* $\_ENV
* $\_COOKIE
* $\_SESSION

|  |  |
| --- | --- |
| $\_SERVER['REQUEST\_METHOD'] | Returns the request method used to access the page  (such as POST) |

$\_SERVER is an array containing information such as headers, paths, and script locations. The entries in this array are created by the web server. There is no guarantee that every web server will provide any of these; servers may omit some, or provide others not listed here.

mysqli\_query — Performs a query on the database

# PHP mysqli\_fetch\_array() Function

Fetch a result row as a numeric array and as an associative array:

## Definition and Usage

The mysqli\_fetch\_array() function fetches a result row as an associative array, a numeric array, or both.

**Note:** Fieldnames returned from this function are case-sensitive.

## Syntax

mysqli\_fetch\_array(result,resulttype);

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| result | Required. Specifies a result set identifier returned by mysqli\_query(), mysqli\_store\_result() or mysqli\_use\_result() |
| resulttype | Optional. Specifies what type of array that should be produced. Can be one of the following values:   * MYSQLI\_ASSOC * MYSQLI\_NUM * MYSQLI\_BOTH |

MYSQL\_ASSOC gives you a relational array, e.g.

Array &#40;

name => Synook,

group => Members

&#41;

# Adding session data

All your session data is stored in the session superglobal array, $\_SESSION, which means that each session variable is one element in that array, combined with its value. Adding variables to this array is done in the same way as adding variables to any array, with the added the bonus that session variables will still be there when your user browses to another page.

To set a session variable, use syntax like this:

$\_SESSION['var'] = $val;  
$\_SESSION['FirstName'] = "Jim";

An alternative way to make data accessible across the various pages of an entire website is to use a PHP Session.

A session creates a file in a temporary directory on the server where registered session variables and their values are stored. This data will be available to all pages on the site during that visit.

The location of the temporary file is determined by a setting in the **php.ini** file called **session.save\_path**. Bore using any session variable make sure you have setup this path.

When a session is started following things happen:

* PHP first creates a unique identifier for that particular session which is a random string of 32 hexadecimal numbers such as 3c7foj34c3jj973hjkop2fc937e3443.
* A cookie called **PHPSESSID** is automatically sent to the user's computer to store unique session identification string.
* A file is automatically created on the server in the designated temporary directory and bears the name of the unique identifier prefixed by sess\_ ie sess\_3c7foj34c3jj973hjkop2fc937e3443.

When a PHP script wants to retrieve the value from a session variable, PHP automatically gets the unique session identifier string from the PHPSESSID cookie and then looks in its temporary directory for the file bearing that name and a validation can be done by comparing both values.

A session ends when the user loses the browser or after leaving the site, the server will terminate the session after a predetermined period of time, commonly 30 minutes duration.

## Starting a PHP Session:

A PHP session is easily started by making a call to the **session\_start()** function.This function first checks if a session is already started and if none is started then it starts one. It is recommended to put the call to **session\_start()** at the beginning of the page.

Session variables are stored in associative array called **$\_SESSION[]**. These variables can be accessed during lifetime of a session.

The following example starts a session then register a variable called **counter** that is incremented each time the page is visited during the session.

Make use of **isset()** function to check if session variable is already set or not.

Put this code in a test.php file and load this file many times to see the result:

|  |
| --- |
| <?php  session\_start();  if( isset( $\_SESSION['counter'] ) )  {  $\_SESSION['counter'] += 1;  }  else  {  $\_SESSION['counter'] = 1;  }  $msg = "You have visited this page ". $\_SESSION['counter'];  $msg .= "in this session.";  ?>  <html>  <head>  <title>Setting up a PHP session</title>  </head>  <body>  <?php echo ( $msg ); ?>  </body>  </html> |

## Destroying a PHP Session:

A PHP session can be destroyed by **session\_destroy()** function. This function does not need any argument and a single call can destroy all the session variables. If you want to destroy a single session variable then you can use **unset()** function to unset a session variable.

Here is the example to unset a single variable:

|  |
| --- |
| <?php  unset($\_SESSION['counter']);  ?> |

Here is the call which will destroy all the session variables:

|  |
| --- |
| <?php  session\_destroy();  ?> |

## Turning on Auto Session:

You don't need to call start\_session() function to start a session when a user visits your site if you can set **session.auto\_start** variable to 1 in **php.ini** file.

## Sessions without cookies:

There may be a case when a user does not allow to store cookies on their machine. So there is another method to send session ID to the browser.

Alternatively, you can use the constant SID which is defined if the session started. If the client did not send an appropriate session cookie, it has the form session\_name=session\_id. Otherwise, it expands to an empty string. Thus, you can embed it unconditionally into URLs.

The following example demonstrates how to register a variable, and how to link correctly to another page using SID.

|  |
| --- |
| <?php  session\_start();  if (isset($\_SESSION['counter'])) {  $\_SESSION['counter'] = 1;  } else {  $\_SESSION['counter']++;  }  ?>  $msg = "You have visited this page ". $\_SESSION['counter'];  $msg .= "in this session.";  echo ( $msg );  <p>  To continue click following link <br />  <a href="nextpage.php?<?php echo htmlspecialchars(SID); >">  </p> |

The **htmlspecialchars()** may be used when printing the SID in order to prevent XSS related attacks.

When you request a web page be brought back to your browser, you're not just bringing back the web page. You're also bringing back something called a HTTP HEADER. This is some extra information, such as type of programme making the request, date requested, should it be displayed as a HTML document, how long the document is, and a lot more besides.

One of the things HTTP HEADER also does is to give status information. This could be whether the page was found (404 errors), and the location of the document. If you want to redirect your users to another page, here's an example:

<?PHP

header("Location: http://www.homeandlearn.co.uk/");

?>

<html>

<body>  
</body>

</html>

Note how the header code goes before any HTML. If you put header code after the HTML, you'll get an error along the lines of "Cannot modify header information."

jQuery is a JavaScript Library.

jQuery greatly simplifies JavaScript programming.

jQuery is easy to learn.

### ID's are unique

* Each element can have only one ID
* Each page can have only one element with that ID

When I was first learning this stuff, I heard over and over that you should only use ID's once, but you can use classes over and over. It basically went in one ear and out the other because it sounded more like a good "rule of thumb" to me rather than something extremely important. If you are purely an HTML/CSS person, this attitude can persist because to you, they really don't seem to do anything different.

Here is one: your code will not pass validation if you use the same ID on more than one element. Validation should be important to all of us, so that alone is a big one.

### Classes are NOT unique

* You can use the same class on multiple elements.
* You can use multiple classes on the same element.

Any styling information that needs to be applied to multiple objects on a page should be done with a class. Take for example a page with multiple "widgets":

<div class="widget"></div>

<div class="widget"></div>

<div class="widget"></div>

You can now use the class name "widget" as your hook to apply the same set of styling to each one of these. But what if you need one of them to be bigger than the other, but still share all the other attributes? Classes has you covered there, as you can apply more than one class:

<div class="widget"></div>

<div class="widget big"></div>

<div class="widget"></div>

No need to make a brand new class name here, just apply a new class right in the class attribute. These classes are space delimited and most browsers support any number of them (actually, it's more like thousands, but way more than you'll ever need).

### Barcodes and Serial Numbers



Maybe a good analogy here is bar codes and serial numbers. Take an iPod in a store. On the packaging will be a bar code. This tells the store what the product is, so when it is scanned, the system knows exactly what the product is and what it costs. It might even be able to know what color it is or where it was kept in the store. All iPod of this same type have the exact same bar code on them.

The iPod will also have a serial number on it which is absolutely unique to any other iPod (or any other device) in the world. The serial number doesn't know the price. It could, but for the store this wouldn't be a very efficient way to store and use that data. Much easier to use the barcode, so that for example, if the price changed, you could just change the price for that bar code and not every individual serial number in your system.

This is much like ID's and Classes. Information that is reusable should be kept in a class and information that is totally unique should be kept in an ID.

### Elements can have BOTH

There is nothing stopping you from having both an ID and a Class on a single element. In fact, it is often a very good idea. Take for example the default markup for a WordPress comment list item:

<li id="comment-27299" class="item">

It has a class applied to it that you may want for styling all comments on the page, but it also has a unique ID value (dynamically generated by WordPress, nicely enough). This ID value is useful for direct linking. Now I can link directly to a particular comment on a particular page easily.

AJAX is about updating parts of a web page, without reloading the whole page.

## What is AJAX?

AJAX = Asynchronous JavaScript and XML.

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.

Examples of applications using AJAX: Google Maps, Gmail, Youtube, and Facebook tabs.

## How AJAX Works



Query parameter

## What About jQuery and AJAX?

jQuery provides several methods for AJAX functionality.

With the jQuery AJAX methods, you can request text, HTML, XML, or JSON from a remote server using both HTTP Get and HTTP Post - And you can load the external data directly into the selected HTML elements of your web page!

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Note** | **Without jQuery, AJAX coding can be a bit tricky!**  Writing regular AJAX code can be a bit tricky, because different browsers have different syntax for AJAX implementation. This means that you will have to write extra code to test for different browsers. However, the jQuery team has taken care of this for us, so that we can write AJAX functionality with only one single line of code.  The jQuery get() and post() methods are used to request data from the server with an HTTP GET or POST request. HTTP Request: GET vs. POST Two commonly used methods for a request-response between a client and server are: GET and POST.   * **GET** - Requests data from a specified resource * **POST** - Submits data to be processed to a specified resource   GET is basically used for just getting (retrieving) some data from the server. **Note:** The GET method may return cached data.  POST can also be used to get some data from the server. However, the POST method NEVER caches data, and is often used to send data along with the request. jQuery $.post() Method The $.post() method requests data from the server using an HTTP POST request.  **Syntax:**  $.post(*URL,data,callback*);  The required URL parameter specifies the URL you wish to request.  The optional data parameter specifies some data to send along with the request.  The optional callback parameter is the name of a function to be executed if the request succeeds.  The following example uses the $.post() method to send some data along with the request: **Definition and Usage** The move\_uploaded\_file() function moves an uploaded file to a new location.  This function returns TRUE on success, or FALSE on failure. **Syntax** move\_uploaded\_file(file,newloc)   |  |  | | --- | --- | | **Parameter** | **Description** | | file | Required. Specifies the file to be moved | | newloc | Required. Specifies the new location for the file | |