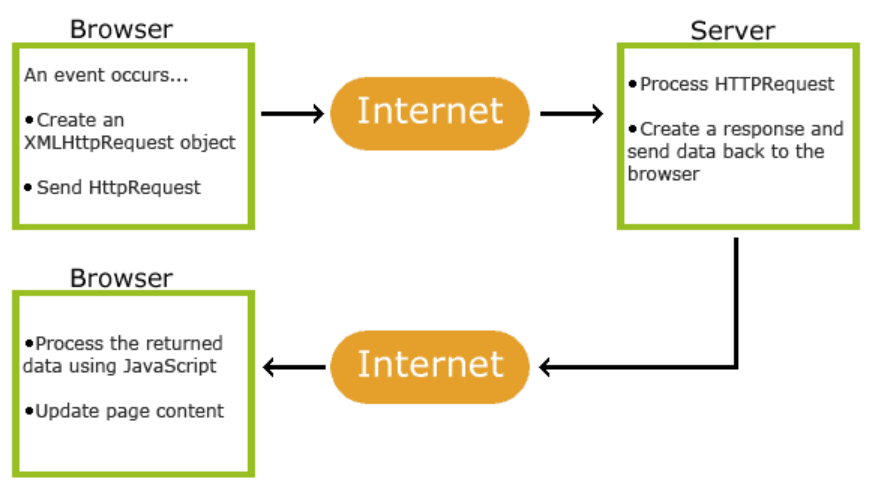
**AJAX (Asynchronous Javascript and XML)**

It just uses a combination of:  
 1. A browser build-in XMLHttpRequest object(to communicate with the server)  
 2. JavaScript and HTML DOM (to make the webpage dynamic)

Features:

1. Update a web page without reloading the page
2. Request data from the server - after the page has loaded
3. Receive data from the server – after the page has loaded
4. Send data to the server – in the background

AJAX workflow:  
 

Steps:

1. To create an XMLHttpRequest,   
    var xmlhttpreq=new XMLHttpRequest();

|  |  |
| --- | --- |
| **Methods** | **Properties** |
| abort() -- to abort the request if it has already sent | readyState -- state of the request 0 unsent; 1 opened; 2 headers\_received; 3 loading; 4 done |
| getAllResponseHeaders() -- returns all the response headers or null | onreadystatechange -- an eventhandler that is called whenever the ready state attribute changes |
| getResponseHeader(headerName) -- returns the specified header value or null | responseType -- defines the response type “text”, “json”, “document”, “blob” , “arraybuffer” |
| open() -- initializes a request | response -- returns a text/json/blob based on responseType |
| send() -- sends the request ---If the request is asynchronous (which is the default), this method returns as soon as the request is sent and the result is delivered using events. If the request is synchronous, this method doesn't return until the response has arrived. | responseText -- returns a DOMString that contains the response or null (if the request is unsuccessful/has not yet been sent) |
| setRequestHeader() -- sets the value of an HTTP request header  details: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers>  details 2: <https://en.wikipedia.org/wiki/List_of_HTTP_header_fields>  details 3: <https://en.wikipedia.org/wiki/Media_type> | status -- status of the response of the request link: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Status>  (**200 OK**, 300 Multiple Choice, 301 Moved Permanently, 302 Found, 304 Not Modified, 307 Temporary Redirect, **400 Bad Request**, 401 Unauthorized, **403 Forbidden**, **404 Not Found**, 408 Bad Request, 410 Gone, **500 Internal Server Error**, 501 Not Implemented, **503 Service Unavailable**) |
|  | statusText -- returns the entire text of the response status message like “200 OK” |

1. To send a GET request,  
    xmlhttpreq.open(“GET”,”ajaxfile.php?var1=value1&val2=value2”,true);  
    xmlhttpreq.send();
2. To send a POST request,  
    xmlhttpreq.open(“POST”,”ajaxfile.php”,true);  
    xmlhttpreq.send(“var1=value1&val2=value2”);

**JSON (JavaScript Object Notation)**

When exchanging data between a browser and a server, the data can only be text. JSON is text that is used to transmit structured data over network connection.  
  
Format:  
1. JSON objects are surrounded by curly braces {}  
2. JSON objects are written in key/value pairs  
3. Keys must be strings and values must be JSON data types(string, number, object, array, boolean or null)  
4. Keys and values are separated by a color  
5. Each key/value pair is separated by a comma

var myobj = {  
 “name” : ”John”,  
 “age” : “30”,  
 “cars” : [  
 {  
 “name” : ”Ford”,  
 “models”: [“fiesta”,”focus”,”mustang”]  
 },  
 {  
 “name”:”BMW”,  
 “models”:[“320”,”x3”,”x5”]  
 },  
 {  
 “name”:”Fiat”,  
 “models”:[“500”,”Panda”]  
 }  
 ]  
};

|  |  |
| --- | --- |
| **Javascript** | **Php** |
| from javascript object to json object,  var obj={name:”John”, age:30, city:”New York”}; var myJSON=JSON.stringify(obj); | from php object to json object,  $arr=array(“a”=>65,”b”=>45,”c”=>35); echo json\_encode($arr); |
| from JSON object to Javascript Object,  var obj = JSON.parse( ‘{ “name” : ”John” , ”age” : 30 , ”city” : ”New York” } ’ ); | from JSON to php object,  $json=’{“a”:65,”b”:45,”c”:35}’; var\_dump(json\_decode($json)); |