AJAX

(https://www.w3schools.com/js/js_ajax_intro.asp)

(https://www.tutorialrepublic.com/javascript-tutorial/javascript-ajax.php)

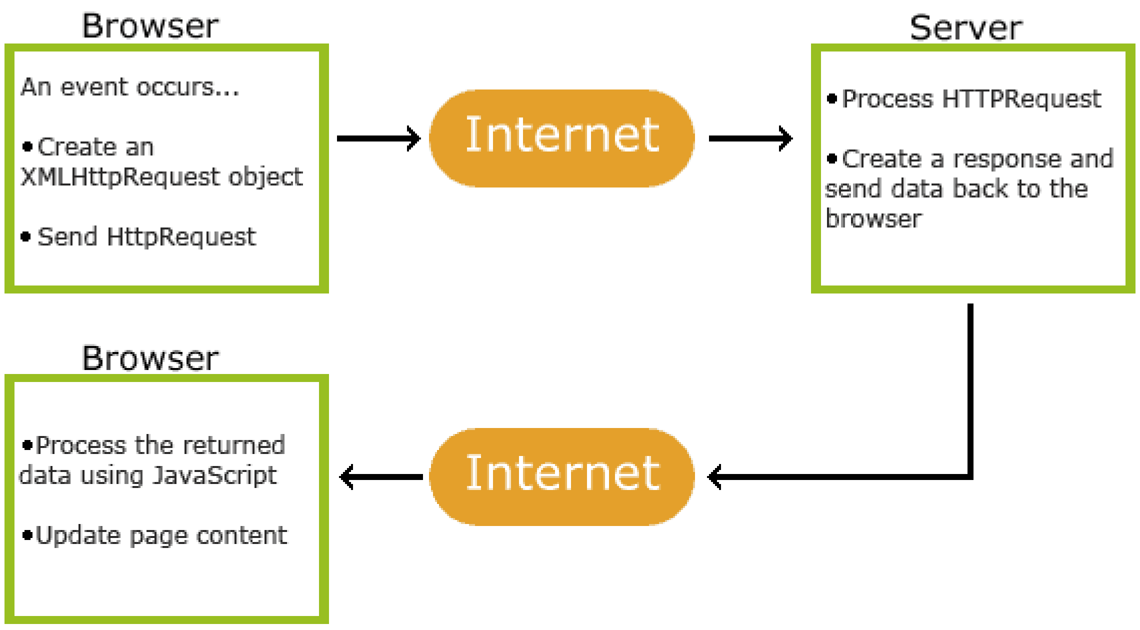
AJAX = **A**synchronous **J**avaScript **A**nd **X**ML.

It is not a programming language. AJAX just uses a combination of:

- A browser built-in `XMLHttpRequest` object (to request data from a web server)
- JavaScript and HTML DOM (to display or use the data)

AJAX allows web pages to be updated asynchronously by exchanging data with a web server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

Ajax has become so popular that you hardly find an application that doesn't use Ajax to some extent. The example of some large-scale Ajax-driven online applications are: Gmail, Google Maps, Google Docs, YouTube, Facebook, and so many other applications.


Browser
An event occurs...
•Create an XMLHttpRequest object
•Send HttpRequest
Internet
Server
•Process HTTPRequest
•Create a response and send data back to the browser
Browser
•Process the returned data using JavaScript
•Update page content
Internet

1. An event occurs in a web page (the page is loaded, a button is clicked)
2. An XMLHttpRequest object is created by JavaScript
3. The XMLHttpRequest object sends a request to a web server
4. The server processes the request
5. The server sends a response back to the web page
6. The response is read by JavaScript
7. Proper action (like page update) is performed by JavaScript

The XMLHttpRequest Object

All modern browsers have a built-in `XMLHttpRequest` object. The `XMLHttpRequest` object can be used to exchange data with a web server behind the scenes. For security reasons, modern browsers do not allow access across domains. This means that both the web page and the XML file it tries to load, must be located on the same server.

Before you perform Ajax communication between client and server, the first thing you must do is to instantiate an `XMLHttpRequest` object, as shown below:

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

Send a Request To a Server

To send a request to a server, we use the open() and send() methods of the `XMLHttpRequest` object:

```
xhttp.open("GET", "ajax_info.txt", true);
xhttp.send();
```

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

GET or POST?

- GET is simpler and faster than POST, and can be used in most cases.
- The GET method is generally used to send small amount of data to the server. Whereas, the POST method is used to send large amount of data, such as form data.
- In GET method, the data is sent as URL parameters. But, in POST method, the data is sent to the server as a part of the HTTP request body. Data sent through POST method will not visible in the URL.
- Sending user input (which can contain unknown characters), POST is more robust and secure than GET.

The url - A File On a Server

The url parameter of the `open()` method, is an address to a file on a server:

```
xhttp.open("GET", "ajax_test.asp", true);
```

The file can be any kind of file, like .txt and .xml, or server scripting files like .asp and .php which can perform actions on the server (e.g. inserting or reading data from database) before sending the response back.

Asynchronous - True or False?

Server requests should be sent asynchronously. The async parameter of the open() method should be set to true: xhttp.open("GET", "ajax_test.asp", true);
By sending asynchronously, the JavaScript does not have to wait for the server response, but can instead: execute other scripts while waiting for server response and deal with the response after the response is ready.

Server Response

The onreadystatechange Property: With the XMLHttpRequest object you can define a function to be executed when the request receives an answer.

- The **readyState** property holds the status of the XMLHttpRequest.
- The **onreadystatechange** property defines a function to be executed when the readyState changes.
- The **status property** and the **statusText property** holds the status of the XMLHttpRequest object.
- The onreadystatechange function is called every time the readyState changes. When **readyState is 4** and **status is 200**, the response is ready.

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

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

Performing an Ajax GET Request

The GET request is typically used to get or retrieve some kind of information from the server that doesn't require any manipulation or change in database, for example, fetching search results based on a term, fetching user details based on their id or name, and so on.

1. A simple `GET` request:

```
xhttp.open("GET", "demo_get.asp", true);
xhttp.send();
```

The following ajax_get_date.html example will show you how to make an Ajax GET request in JavaScript.

```
<!DOCTYPE html>
<html>
<head> <title>JavaScript Ajax GET Demo</title>
<body>
   <script>
   function fetchDoc() {
      var httpRequest = new XMLHttpRequest();
      httpRequest.onreadystatechange = function() {
         if (this.readyState === 4 && this.status === 200) {
            document.getElementById("output").innerHTML = this.responseText;
         }
      };
      httpRequest.open("GET", "ajax_get_date.php", true);
      httpRequest.send();
    }
    </script>
    <button type="button" onclick="fetchDoc()">Fetch Content</button>
    <div id="output"></div>
</body>
</html>
```

Here's the code from "ajax_get_date.php" file that simply outputs the present date.

```
<?php
echo date("d/m/Y");
?>
```

** Note: In order to execute these programms:*

- *Is your localhost working? can you access it via http://localhost*

  If not, install a local server via: https://www.apachefriends.org/index.html
- *Now, create your html and php files; place it in the "htdocs" folder of XAMPP, which is likely to have its folder on your C: hard drive path, unless you moved it elsewhere.*
- *Start XAMPP Apache server. Now you can access your html web page via web browser at "localhost/program.html".*

2. If you want to send information with the `GET` method, add the information to the URL:

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

The following get_welcome.html example shows how to send information with the `GET` method:

```
<!DOCTYPE html>
<head>
<script>
function displayFullName()
{
    var request = new XMLHttpRequest();
    request.open("GET", "get_welcome.php?fname=John&lname=Clark",true);
    request.onreadystatechange = function()
    {
        if(this.readyState === 4 && this.status === 200)
        {
            document.getElementById("result").innerHTML = this.responseText;
        }
    };
    request.send();
}
</script>
</head>
<body>
   <p id="result">This content will be replaced by the server response</p>
    <button type="button" onclick="displayFullName()">Display Full Name</button>
</body>
</html>
```

Here's the code from "get_welcome.php" file that simply creates the full name of a person by joining their first name and last name and outputs a greeting message.

```
<?php
if(isset($_GET["fname"]) && isset($_GET["lname"]))
{
$fname = htmlspecialchars($_GET["fname"]);
$lname = htmlspecialchars($_GET["lname"]);
$fullname = $fname . " " . $lname;

echo "Hello, $fullname! Welcome to our website.";
}
else
{  echo "Hi there! Welcome to our website.";
}
?>
```

Performing an Ajax POST Request

The POST method is mainly used to submit a form data to the web server.

1. A simple POST request:

```
xhttp.open("POST", "demo_post.asp", true);
xhttp.send();
```

The following ajax_post_date.html example will show you how to make an Ajax POST request in JavaScript.

```
<!DOCTYPE html>
<html>
<head> <title>JavaScript Ajax POST Demo</title></head>
<body>
<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML = this.responseText;
    }
  };
  xhttp.open("POST", "ajax_date.php", true);
  xhttp.send();
}
</script>

<button type="button" onclick="loadDoc()">Request date</button>
<p id="demo">Result</p>
</body>
</html>
```

Here's the code from "ajax_date.php" file that simply outputs the present date.

```
<?php

echo date("d/m/Y");

?>
```

2. To POST data like an HTML form, add an HTTP header with `setRequestHeader()`. Specify the data you want to send in the `send()` method:

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

In GET method, the data is sent as URL parameters. But, in POST method, the data is sent to the server as a part of the HTTP request body. Data sent through POST method will not visible in the URL. The `send()` method accepts an optional *body* parameter which allow us to specify the request's body. This is primarily used for HTTP POST requests.

Method	Description
setRequestHeader(*header, value*)	Adds HTTP headers to the request ***header*:** specifies the header name ***value*:** specifies the header value

Following example shows how to submit form data using POST method:

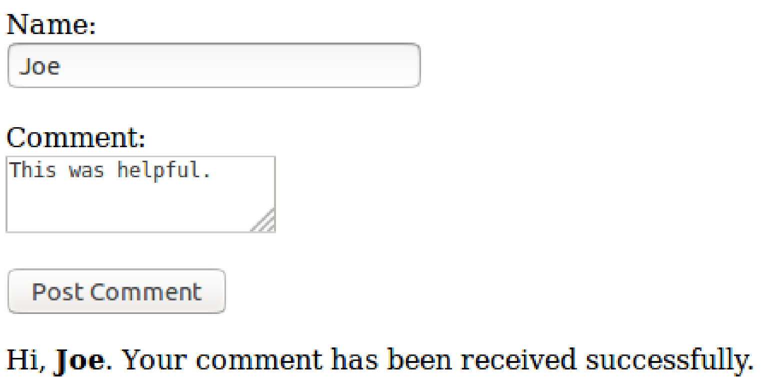
Here is the html code: post_confirmation.html

```
<!DOCTYPE html>
<head>
<script>
function postComment()
{
    var request = new XMLHttpRequest();
    request.open("POST", "post_confirmation.php",true);
    request.onreadystatechange = function()
    {
        if(this.readyState === 4 && this.status === 200)
        {
            document.getElementById("result").innerHTML = this.responseText;
        }
    };
    request.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
    var name = document.getElementById("name").value;
    var comment = document.getElementById("comment").value;
    var formdata="name=" + name + "&comment=" + comment;
    request.send(formdata);
}
</script>
</head>
<body>
    <form id="myForm">
       Name:  <div><input type="text" name="name" id="name"></div><br>
       Comment: <div><textarea name="comment" id="comment"></textarea></div><br>
       <button type="button" onclick="postComment()">Post Comment</button>
    </form>
        <p id="result">This content will be replaced by the server response</p>
</body>
</html>
```

Here's the code of "post_confirmation.php" file that simply outputs the values submitted by the user.

- “**htmlspecialchars**” Convert special characters to HTML entities.
- **“Trim”**: is to make sure it isn't empty which it is after trim on a white space only name

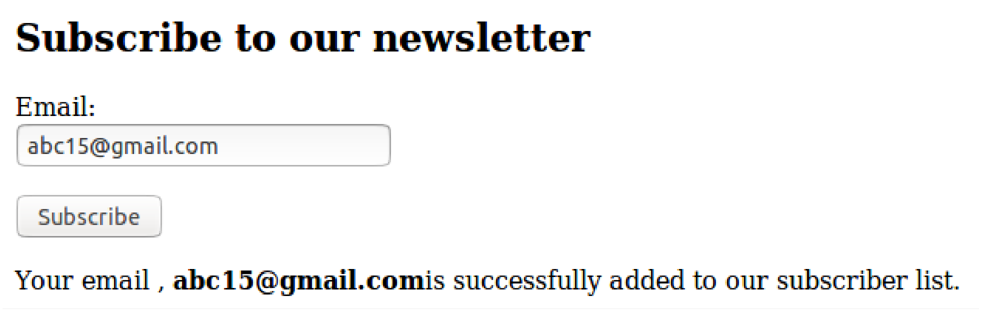
```php
<?php
if($_SERVER["REQUEST_METHOD"] == "POST")
{
  $name = htmlspecialchars(trim($_POST["name"]));
  $comment = htmlspecialchars(trim($_POST["comment"]));

  if(!empty($name) && !empty($comment))
  {
     echo "<p>Hi, <b>$name</b>. Your comment has been received successfully.<p>";
  }
  else
  { echo "<p>Please fill all the fields in the form!</p>";
  }
} else
  { echo "<p>Something went wrong. Please try again.</p>";
  }
?>
```



Hi, **Joe**. Your comment has been received successfully.

More Examples:

1. Following example shows a message with data entered by the used on clicking the button:



Subscribe to our newsletter

Email:

abc15@gmail.com

Your email , **abc15@gmail.com**is successfully added to our subscriber list.

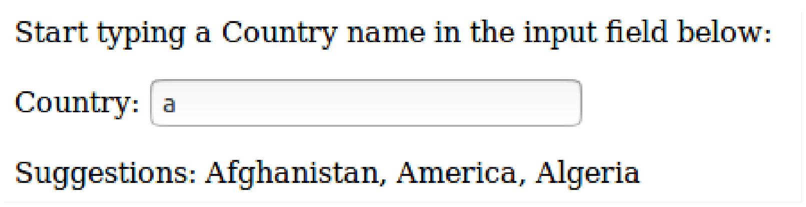
Here is the post_email.html code for the above example:

```
<!DOCTYPE html>
<head>
<script>
function addSubscriber()
{
    var request = new XMLHttpRequest();
    request.open("POST", "post_email.php");
    request.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
        request.onreadystatechange = function()
        {
           if(this.readyState === 4 && this.status === 200)
          {
            document.getElementById("result").innerHTML = this.responseText;
          }
       };
       var email = document.getElementById("email").value;
       request.send("email=" + email);
}
</script>
</head>
<body>
    <form id="myForm">
        <h2>Subscribe to our newsletter</h2>
        Email: <div><input type="text" id="email"></div>
        <p><button type="button" onclick="addSubscriber()">Subscribe</button></p>
    </form>
        <p id="result">This content will be replaced by the server response</p>
</body>
</html>
```

Here's the code of "post_email.php" file that simply outputs the values submitted by the user.

```
<?php
if($_SERVER["REQUEST_METHOD"] == "POST")

  $email = htmlspecialchars(trim($_POST["email"]));

  if(!empty($email))
  {
  echo "<p> Your email , <b>$email</b>is successfully added to our subscriber list.<p>";
  }
?>
```

2. The following example demonstrates how a web page can communicate with a web server while a user types characters in an input field: (https://www.w3schools.com/js/js_ajax_php.asp)


Start typing a Country name in the input field below:

Country: a

Suggestions: Afghanistan, America, Algeria

In the example, when a user types a character in the input field, a function called `showHint()` is executed. The function is triggered by the `onkeyup` event. Here is the HTML code: get_country.html

```
<!DOCTYPE html>
<html>
<body>
<div>
  <p>Start typing a Country name in the input field below:</p>
  <p>Country: <input type="text" onkeyup="showHint(this.value)"  ></p>
  <p>Suggestions:  <span id="result"></span> </p>
</div>

<script>
function showHint(name)
{
  if (name.length >= 0)
  {
     var xhttp = new XMLHttpRequest();
     xhttp.onreadystatechange = function()
     {
       if (this.readyState === 4 && this.status === 200)
       {
          document.getElementById("result").innerHTML = this.responseText;
       }
     };
     xhttp.open("GET", "get_country.php?q=" + name, true);
     xhttp.send();
  }
}
</script>
</body>
</html>
```

The PHP file checks an array of names, and returns the corresponding name(s) to the browser:

- stristr() Finds the first occurrence of a string inside another string (case-insensitive)
- substr() Returns a part of a string, it has three parameters. The first is the string, the second is where to start from and the third is the number of characters.
- example: substr('abcdef', 0, 4); gives output: abcde

PHP File - "get_country.php"

```php
<?php

if (isset($_GET["q"]))
{
 $countries = array("Afghanistan","America", "Algeria","Canada", "India", "Nepal");
 $Text = $_GET["q"] ;
 $hint = "";

 if ($Text !== "")   //Loop through all hints from array if $Text is not empty.
 {
   foreach($countries as $name)
   {
      if (stristr($Text, substr($name, 0, strlen($Text))))
      {
         if ($hint === "")
         {
            $hint = $name;
         }
         else
         {
             $hint .= ", $name";   //for more than one name from silimar alphabet
         }
      }
   }
 }
// Output "no suggestion" if no hint was found or output correct values
//"? :" operator is used

echo $hint === " " ? "no suggestion" : $hint;
}
?>
```