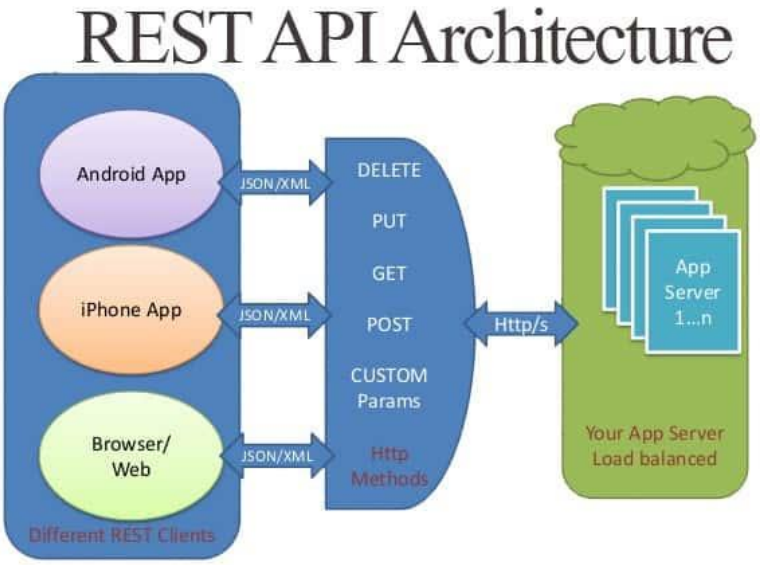
What is the REST API & How it works

REST stands for Representational State Transfer, REST API used for creating web services that can be accessed on different applications and created with CRUD (Create, Read, Update, Delete) operations. REST uses HTTP methods like GET, POST, PUT and DELETE to perform operations.

- GET – To retrieve information.
- POST – To create a new record.
- PUT – To update existing records.
- DELETE – To delete records.

In the REST API, we can get responses in JSON or XML format but we will use the JSON format because it is lightweight and easy to perform the parsing. Here, we will create a REST API to get customer data by passing the customer id in PHP.



Steps to create a REST API in PHP with MySQL

1. Create database table
2. Connect database
3. Create a file for REST API
4. Rewrite the API URL
5. Output

Prof. Ankita Chauhan
Assistant Professor
CE Department
MBIT

1. Create database table

We will create a customer table for a small example and for that we have created a `demo` database in MySQL. Run the following script to create a table in the database.

```
CREATE TABLE customers(
 customer_id int(11) NOT NULL PRIMARY KEY AUTO_INCREMENT,
 customer_name varchar(255) NOT NULL,
 customer_email varchar(255) NOT NULL,
 customer_contact varchar(255) NOT NULL,
 customer_address varchar(255) NOT NULL,
 country varchar(255) NOT NULL
) ENGINE=InnoDB;
```

You can insert some sample records to the table for REST API testing.

2. Connect database

Now, we have to create a `connection.php` file and add the following code to connect the database.

connection.php

```
<?php
 $dbhost = "localhost";
 $dbuser = "root";
 $dbpass = "";
 $db = "demo";
 $con = mysqli_connect($dbhost, $dbuser, $dbpass , $db) or die($con);
?>
```

3. Create a file for REST API

In the next step, we will create an `api.php` file at the root level of the directory to create a REST API and add the following code in that file.

api.php

```
<?php
```

```
header("Content-Type:application/json");
include('connection.php');

if (isset($_GET['customer_id']) && $_GET['customer_id']!="") {

 $customer_id = $_GET['customer_id'];
 $query = "SELECT * FROM `customers` WHERE customer_id=$customer_id";
 $result = mysqli_query($con,$query);
 $row = mysqli_fetch_array($result,MYSQLI_ASSOC);

 $customerData['customer_id'] = $row['customer_id'];
 $customerData['customer_name'] = $row['customer_name'];
 $customerData['customer_email'] = $row['customer_email'];
 $customerData['customer_contact'] = $row['customer_contact'];
 $customerData['customer_address'] = $row['customer_address'];
 $customerData['country'] = $row['country'];

 $response["status"] = "true";
 $response["message"] = "Customer Details";
 $response["customers"] = $customerData;

} else {
 $response["status"] = "false";
 $response["message"] = "No customer(s) found!";
}
echo json_encode($response); exit;

?>
```

Now we can make an HTTP GET request and get the customer data by passing the customer id. You can run the following link to get the customer data.

http://*localhost/demo/api.php?customer_id=1*

Output:

{"status":"true","message":"Customer Details","customers":{"customer_id":"1","customer_name":"Clue Mediator","customer_email":"contact@cluemediator.com","customer_contact":"9998887776","customer_address":"Address","country":"US"}}

4. Rewrite the API URL

Now, we need to rewrite the above URL using .htaccess file because it's not user friendly.
.htaccess

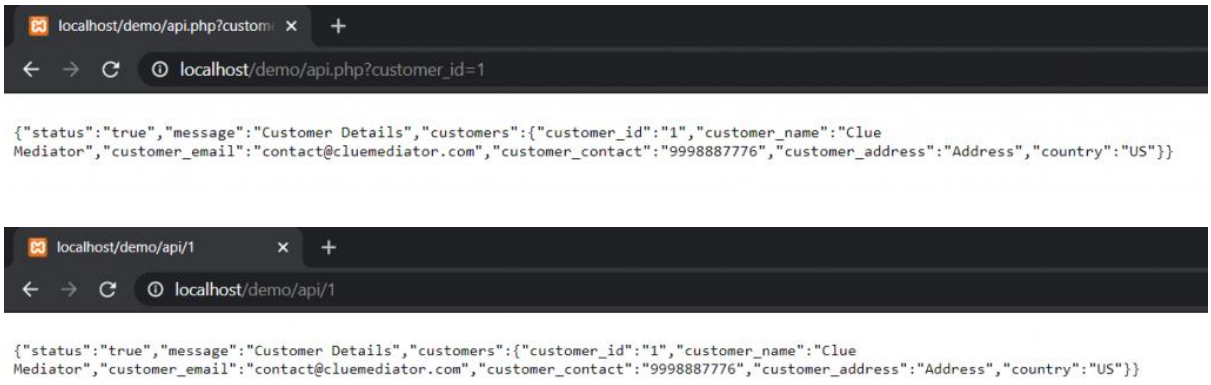
```
1RewriteEngine On
2RewriteRule ^api/([0-9a-zA-Z_-]*)$ api.php?customer_id=$1 [NC,L]
```

After adding the above code in `.htaccess` file, we can retrieve the customer data by browsing the following URL and get the same output.

http://*localhost/demo/api/1*

Output

Run the above given URLs in the browser to check the GET API.



What is AJAX?

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

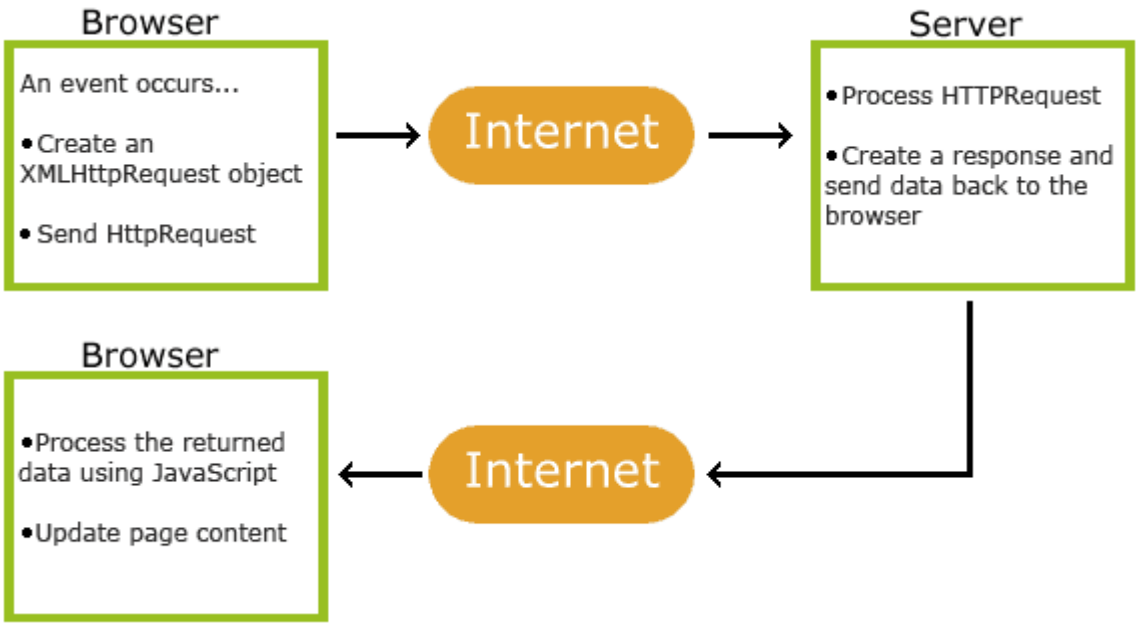
AJAX is not a programming language.

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.

How AJAX Works



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

AJAX Example

```
<!DOCTYPE html>

<html>

<body>

<div id="demo">
```

```
<h2>The XMLHttpRequest Object</h2>

<button type="button" onclick="loadDoc()">Change Content</button>

</div>

<script>

function loadDoc() {

  const xhttp = new XMLHttpRequest();

  xhttp.onload = function() {

    document.getElementById("demo").innerHTML =

    this.responseText;

  }

  xhttp.open("GET", "ajax_info.txt");

  xhttp.send();

}

</script>

</body>

</html>
```

The HTML page contains a section and a <button>.

The section is used to display information from a server.

The <button> calls a function (if it is clicked).

The function requests data from a web server and displays it:

jQuery

The purpose of jQuery is to make it much easier to use JavaScript on your website.

Prof. Ankita Chauhan
Assistant Professor
CE Department
MBIT

What You Should Already Know

Before you start studying jQuery, you should have a basic knowledge of:

- HTML
- CSS
- JavaScript

What is jQuery?

jQuery is a lightweight, "write less, do more", JavaScript library.

The purpose of jQuery is to make it much easier to use JavaScript on your website.

jQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish, and wraps them into methods that you can call with a single line of code.

jQuery also simplifies a lot of the complicated things from JavaScript, like AJAX calls and DOM manipulation.

The jQuery library contains the following features:

- HTML/DOM manipulation
- CSS manipulation
- HTML event methods
- Effects and animations
- AJAX
- Utilities

Why jQuery?

There are lots of other JavaScript libraries out there, but jQuery is probably the most popular, and also the most extendable.

Many of the biggest companies on the Web use jQuery, such as:

- Google
- Microsoft
- IBM
- Netflix

Adding jQuery to Your Web Pages

There are several ways to start using jQuery on your web site. You can:

- Download the jQuery library from jQuery.com
- Include jQuery from a CDN, like Google

Downloading jQuery

There are two versions of jQuery available for downloading:

- Production version - this is for your live website because it has been minified and compressed
- Development version - this is for testing and development (uncompressed and readable code)

Both versions can be downloaded from jQuery.com.

The jQuery library is a single JavaScript file, and you reference it with the HTML <script> tag (notice that the <script> tag should be inside the <head> section):

```
<head>
<script src="jquery-3.6.0.min.js"></script>
</head>
```

jQuery CDN

If you don't want to download and host jQuery yourself, you can include it from a CDN (Content Delivery Network).

Google is an example of someone who host jQuery:

Google CDN:

```
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
</head>
```

jQuery Syntax

The jQuery syntax is tailor-made for **selecting** HTML elements and performing some **action** on the element(s).

Basic syntax is: **$(*selector*).*action*()**

- A $ sign to define/access jQuery
- A (*selector*) to "query (or find)" HTML elements
- A jQuery *action*() to be performed on the element(s)

Examples:

- $(this).hide() - hides the current element.

- $("p").hide() - hides all <p> elements.
- $(".test").hide() - hides all elements with class="test".
- $("#test").hide() - hides the element with id="test".

The Document Ready Event

You might have noticed that all jQuery methods in our examples, are inside a document ready event:

```
$(document).ready(function(){

  // jQuery methods go here...

});
```

jQuery Selectors

jQuery selectors allow you to select and manipulate HTML element(s).

jQuery selectors are used to "find" (or select) HTML elements based on their name, id, classes, types, attributes, values of attributes and much more. It's based on the existing CSS Selectors, and in addition, it has some own custom selectors.

All selectors in jQuery start with the dollar sign and parentheses: $().

The element Selector

The jQuery element selector selects elements based on the element name.

You can select all <p> elements on a page like this:

```
$("p")
```

The element Selector

The jQuery element selector selects elements based on the element name.

You can select all <p> elements on a page like this:

```
$("p")
```

The #id Selector

The jQuery *#id* selector uses the id attribute of an HTML tag to find the specific element.

An id should be unique within a page, so you should use the #id selector when you want to find a single, unique element.

To find an element with a specific id, write a hash character, followed by the id of the HTML element:

```
$("#test")
```

Example

When a user clicks on a button, the element with id="test" will be hidden:

Example

```
$(document).ready(function(){
  $("button").click(function(){
    $("#test").hide();
  });
});
```

The .class Selector

The jQuery *.class* selector finds elements with a specific class.

To find elements with a specific class, write a period character, followed by the name of the class:

```
$(".test")
```

Example

When a user clicks on a button, the elements with class="test" will be hidden:

Example

```
$(document).ready(function(){
  $("button").click(function(){
    $(".test").hide();
  });
});
```

Example:

```
<!DOCTYPE html>

<html>

<head>
```

```
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

<script>

$(document).ready(function(){

  $("p").click(function(){

    $(this).hide();

  });

});

</script>

</head>

<body>

<p>If you click on me, I will disappear.</p>

<p>Click me away!</p>

<p>Click me too!</p>

</body>

</html>
```

jQuery Syntax For Event Methods

In jQuery, most DOM events have an equivalent jQuery method.

To assign a click event to all paragraphs on a page, you can do this:

```
$("p").click();
```

The next step is to define what should happen when the event fires. You must pass a function to the event:

```
$("p").click(function(){
  // action goes here!!
});
```

Commonly Used jQuery Event Methods

$(document).ready()

The $(document). ready() method allows us to execute a function when the document is fully loaded. This event is already explained in the jQuery Syntax chapter.

click ()

The click() method attaches an event handler function to an HTML element.

The function is executed when the user clicks on the HTML element.

The following example says: When a click event fires on a <p> element; hide the current <p> element:

Example

```
$("p").click(function(){
  $(this).hide();
});
```

dblclick()

The dblclick() method attaches an event handler function to an HTML element.

The function is executed when the user double-clicks on the HTML element:

Example

```
$("p").dblclick(function(){
  $(this).hide();
});
```

mouseenter()

The mouseenter() method attaches an event handler function to an HTML element.

The function is executed when the mouse pointer enters the HTML element:

Example

```
$("#p1").mouseenter(function(){
  alert("You entered p1!");
});
```

mouseleave()

The mouseleave() method attaches an event handler function to an HTML element.

The function is executed when the mouse pointer leaves the HTML element:

Example

```
$("#p1").mouseleave(function(){
  alert("Bye! You now leave p1!");
});
```

mousedown()

The mousedown() method attaches an event handler function to an HTML element.

The function is executed, when the left, middle or right mouse button is pressed down, while the mouse is over the HTML element:

Example

```
$("#p1").mousedown(function(){
  alert("Mouse down over p1!");
});
```

mouseup()

The mouseup() method attaches an event handler function to an HTML element.

The function is executed, when the left, middle or right mouse button is released, while the mouse is over the HTML element:

Example

```
$("#p1").mouseup(function(){
  alert("Mouse up over p1!");
});
```

hover()

The hover() method takes two functions and is a combination of the mouseenter() and mouseleave() methods.

The first function is executed when the mouse enters the HTML element, and the second function is executed when the mouse leaves the HTML element:

Example

```
$("#p1").hover(function(){
  alert("You entered p1!");
},
function(){
```

```
  alert("Bye! You now leave p1!");
});
```

focus()

The focus() method attaches an event handler function to an HTML form field.

The function is executed when the form field gets focus:

Example

```
$("input").focus(function(){
  $(this).css("background-color", "#cccccc");
});
```

blur()

The blur() method attaches an event handler function to an HTML form field.

The function is executed when the form field loses focus:

Example

```
$("input").blur(function(){
  $(this).css("background-color", "#ffffff");
});
```