Universidad Nacional de San Agustín

Facultad de Ingeniería de Producción y Servicios

Escuela Profesional de Ingeniería de Sistemas

Segundo Año

Lab 3 - AJAX

**Curso:**

Programación Web 2

**Alumno:**

Ramirez Ccahuana Max Edú

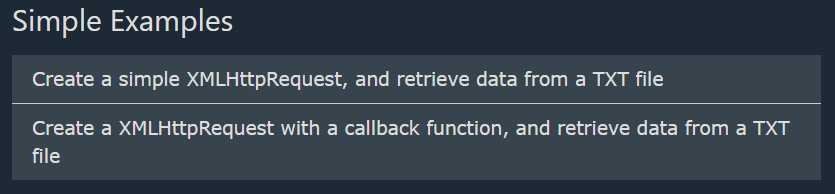
**Docente:**

Ing. Carlo José Luis Corrales Delgado

**Arequipa - 2023**

**Informe de Ejemplos de w3schools**

Para todos los ejemplos se estará usando el servidor local de python en el puerto 8000. A modo de práctica, no se usarán los mismos datos de los ejemplos, pero sí se seguirá el objetivo principal del ejercicio.



**Ejemplo 1**

index.html recupera datos de un archivo example1.txt al hacer click en un botón “Cargar datos”

<!DOCTYPE html>

<html>

<head>

    <title>Prueba AJAX</title>

</head>

<body>

    <h1>Prueba AJAX</h1>

    <button onclick=test()>Cargar Datos</button>

    <div id="box"></div>

    <script>

        function test(){

            const xhttp = new XMLHttpRequest();

            xhttp.onload = function() {

                document.getElementById("box").innerHTML = this.responseText;

            }

            xhttp.open("GET", "example1.txt", true);

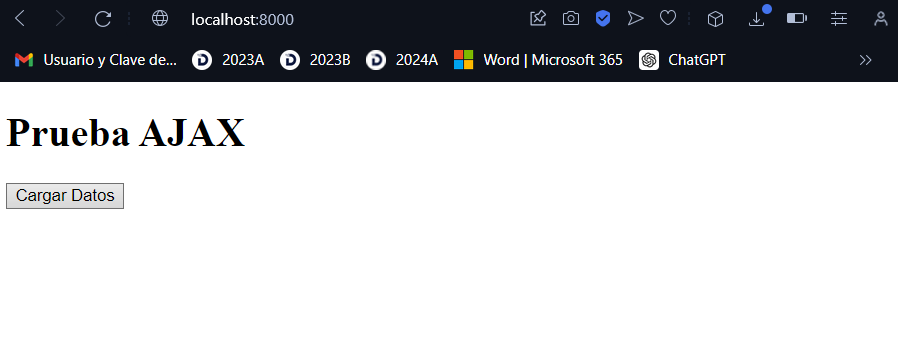
            xhttp.send();

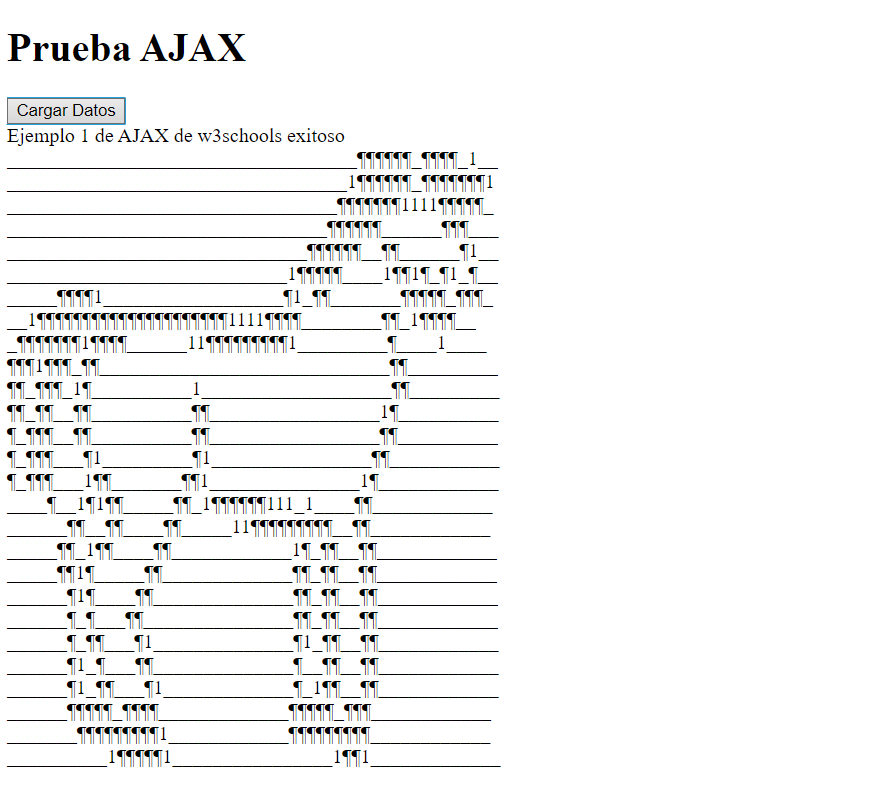
        }

    </script>

</body>

</html>





**Ejemplo 2**

Similar al primer ejercicio, pero usando una funcion callback, que permite la continuidad del programa incluso si esta no se ha completado.

<!DOCTYPE html>

<html>

<head>

    <title>Prueba AJAX 2</title>

</head>

<body>

    <h1>Prueba AJAX 2</h1>

    <div>

        <button onclick = test("text1.txt")>Primer texto</button>

        <button onclick = test("text2.txt")>Segundo texto</button>

    </div>

    <div id="box"></div>

    <script>

        function test(url){

            const xhttp = new XMLHttpRequest();

            xhttp.onload = function() {

                auxFunction(xhttp);

            }

            xhttp.open("GET", url, true);

            xhttp.send();

        }

        function auxFunction(xhttp){

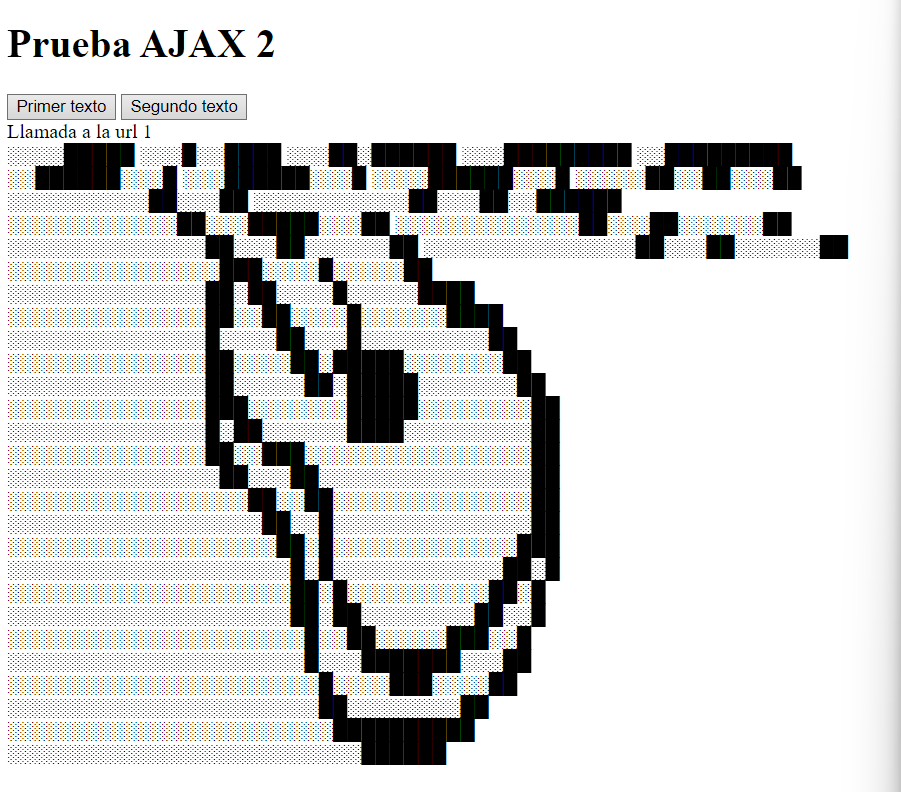
            document.getElementById("box").innerHTML = xhttp.responseText;

        }

    </script>

</body>

</html>

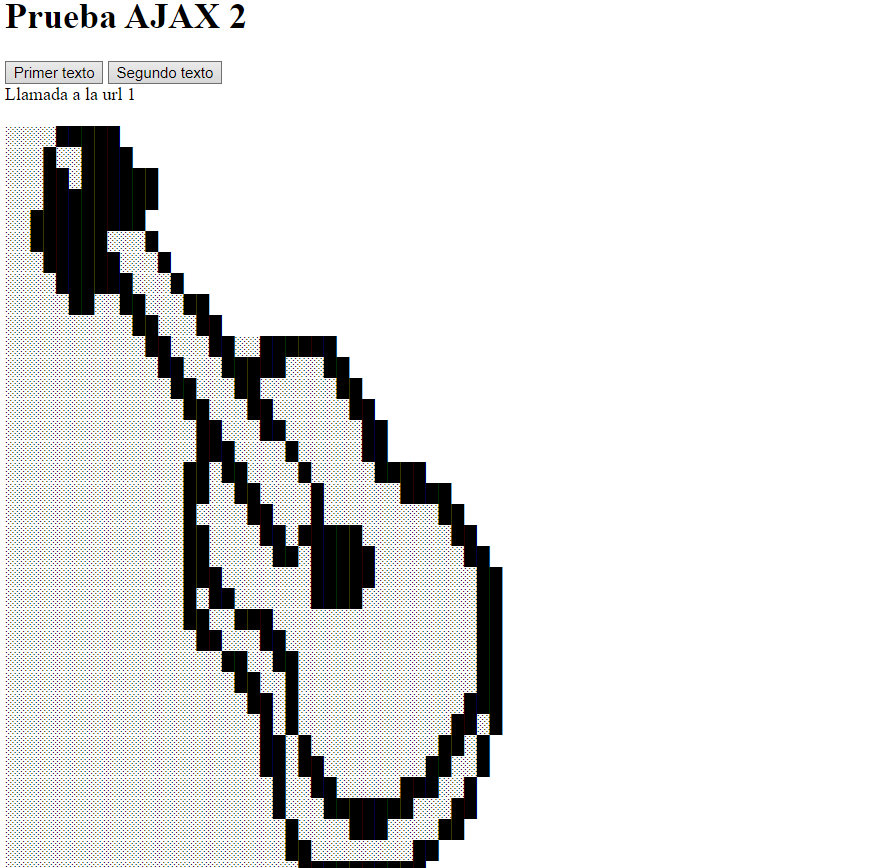


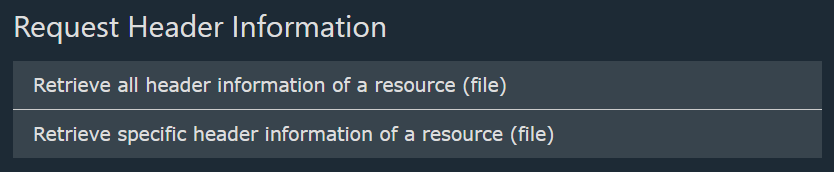
Aquí surge un problema, pues el texto no se muestra correctamente. Sin embargo, esto se soluciona fácilmente al agregar al texto de respuesta:

.replace(/\n/g, '<br>')

Así, reemplaza los saltos de línea “\n” en el txt a etiquetas “<br>” que la página html puede interpretar correctamente.

Ahora, el texto se muestra correctamente:





**Ejemplo 3**

Para este ejemplo, usamos la función getAllResponseHeaders(), que devuelve todos los valores en el header del documento que solicitamos, en otras palabras, “metadatos”.

<!DOCTYPE html>

<html>

<head>

    <title>Prueba AJAX 3</title>

</head>

<body>

    <h1>Prueba AJAX 3</h1>

    <div>

        <button onclick = getHeader()>Solicitar header del archivo</button>

    </div>

    <div id="box"></div>

    <script>

        function getHeader(){

            const xhttp = new XMLHttpRequest();

            xhttp.onload = function() {

                document.getElementById("box").innerHTML = this.getAllResponseHeaders();

            }

            xhttp.open("GET", "text.txt", true);

            xhttp.send();

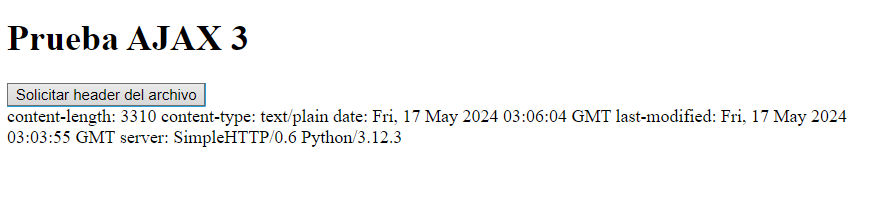
        }

    </script>

</body>

</html>





**Ejemplo 4**

Para este ejemplo, solicitamos también metadatos del archivo, pero esta vez especificando qué campo deseamos. Para comparaciones, se hizo un botón para todos los metadatos y otro para solamente la fecha de última modificación:

<!DOCTYPE html>

<html>

<head>

    <title>Prueba AJAX 4</title>

    <meta charset = "utf-8">

</head>

<body>

    <h1>Prueba AJAX 4</h1>

    <div>

        <button onclick = getHeader("allHeaders")>Solicitar header completo del archivo</button>

        <button onclick = getHeader("lastModification")>Solicitar última modificación del archivo</button>

    </div>

    <div id="box"></div>

    <script>

        function getHeader(type){

            const xhttp = new XMLHttpRequest();

            xhttp.onload = function() {

                var element = document.getElementById("box");

                if(type == "allHeaders"){

                    element.innerHTML = this.getAllResponseHeaders();

                }else if (type == "lastModification"){

                    element.innerHTML = "Ultima modificacion:" + this.getResponseHeader("Last-Modified");

                }

            }

            xhttp.open("GET", "text.txt", true);

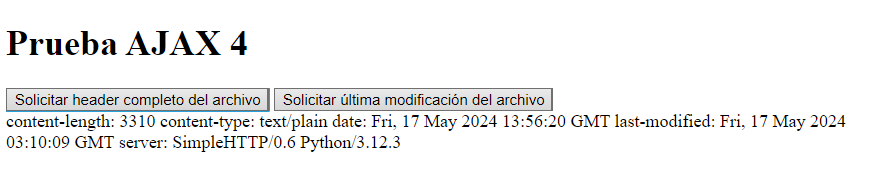
            xhttp.send();

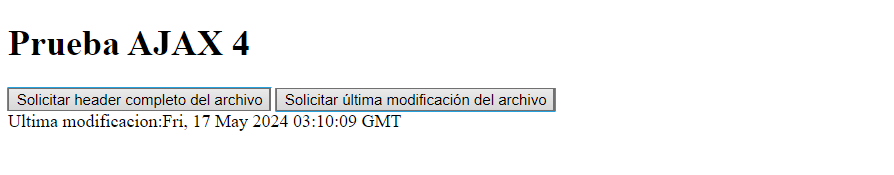
        }

    </script>

</body>

</html>





**Ejemplo 5**

En este ejemplo, se solicitan datos de un archivo “example.xml”, como su código de estado, su estado en texto y su contenido:

<!DOCTYPE html>

<html>

<head>

    <title>Prueba AJAX 5</title>

    <meta charset = "utf-8">

</head>

<body>

    <h1>Prueba AJAX 5</h1>

    <div>

        <button onclick = loadData()>Cargar datos</button>

     </div>

    <p><b>Status: </b><div id="statusBox"></div></p>

    <p><b>Status Text: </b><div id="statusTextBox"></div></p>

    <p><b>Response Text: </b><div id="responseBox"></div></p>

    <script>

        function loadData(){

            const xhttp = new XMLHttpRequest();

            xhttp.onload = function() {

                document.getElementById("statusBox").innerHTML = this.status;

                document.getElementById("statusTextBox").innerHTML = this.statusText;

                document.getElementById("responseBox").innerHTML = this.responseText;

            }

            xhttp.open("GET", "example.xml", true);

            xhttp.send();

        }

    </script>

</body>

</html>





**Ejemplo 6**

Aquí, leemos los datos de un archivo “data.xml” y en base a sus etiquetas, los ordenamos en una tabla:

<!DOCTYPE html>

<html>

<style>

    table, th, td{

        border : 1px solid black;

        border-collapse: collapse;

    }

</style>

<head>

    <title>Prueba AJAX 6</title>

    <meta charset = "utf-8">

</head>

<body>

    <h1>Prueba AJAX 6</h1>

    <div>

        <button onclick = loadData()>Cargar datos</button>

    </div>

    <br>

    <table id = "dataTable"></table>

    <script>

        function loadData(){

            const xhttp = new XMLHttpRequest();

            xhttp.onload = function() {

                generateTable(this);

            }

            xhttp.open("GET", "data.xml", true);

            xhttp.send();

        }

        function generateTable(xml){

            const xmlDoc = xml.responseXML;

            const x = xmlDoc.getElementsByTagName("data");

            var table = "<tr><th>Campo</th><th>Valor</th></tr>";

            for (let i = 0; i < x.length; i++){

                table += "<tr><td>" +

                    x[i].getElementsByTagName("field")[0].childNodes[0].nodeValue + "</td><td>" +

                    x[i].getElementsByTagName("value")[0].childNodes[0].nodeValue + "</td><tr>";

            }

            document.getElementById("dataTable").innerHTML = table;

        }

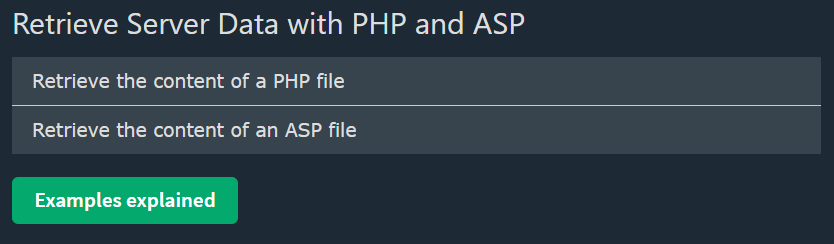
    </script>

</body>

</html>







**Ejemplo 7**

En este ejemplo, usamos php para procesar una solicitud cada que se ejecuta la función onkeyup = predict() en un <input> de index.html

<!DOCTYPE html>

<html>

<head>

    <title>Prueba AJAX 7</title>

    <meta charset = "utf-8">

</head>

<body>

    <h1>Prueba AJAX 7</h1>

    <h2>Escriba un nombre debajo:</h2>

    <p>Posibles predicciones: <span id="predictions"></span></p>

    <p>Nombre: <input type="text" id="nameBox" onkeyup=predict(this.value)></p>

    <br>

    <script>

        function predict(str){

            if(str.length == 0){

                document.getElementById("predictions").innerHTML = "Ingrese un nombre";

                return;

            }

            const xhttp = new XMLHttpRequest();

            xhttp.onload = function() {

                console.log(this.response);

                document.getElementById("predictions").innerHTML = this.response;

            }

            xhttp.open("GET", "suggestions.php?q=" + str, true);

            xhttp.send();

        }

    </script>

</body>

</html>

<?php

$names = ["Isaac", "Magdalene", "Cain", "Judas", "Blue", "Eve", "Samson", "Azazel", "Lazarus", "Eden", "Lost", "Lilith", "Keeper", "Apollyon", "Forgotten", "Bethany", "Jacob", "Esau"];

$q = isset($\_GET['q']) ? $\_GET['q'] : '';

$predict = "";

if ($q !== "") {

    $q = strtolower($q);

    foreach ($names as $name) {

        if (stristr(strtolower($name), $q)) {

            if ($predict === "") {

                $predict = $name;

            } else {

                $predict .= ", $name";

            }

        }

    }

}

echo $predict === "" ? "no hay coincidencias" : $predict;

?>

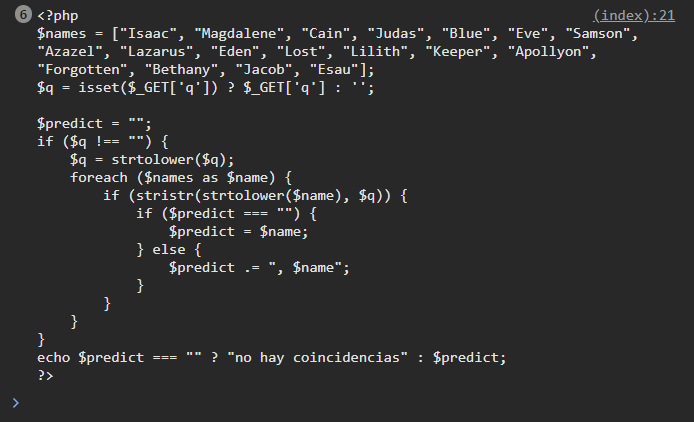
Sin embargo, al probarlo en <http://localhost:8000> no se ejecuta correctamente.





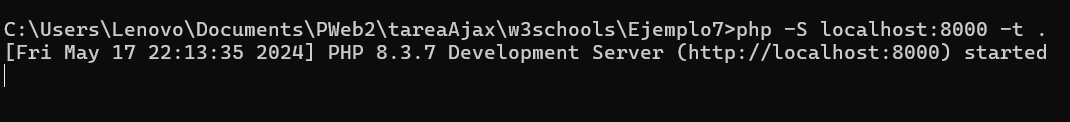
La función predict() parece estar ejecutándose correctamente, pues carga “Ingrese un nombre” al dejar el input vacío.

Revisando la consola de Javascript, nos damos cuenta de que el archivo php no está siendo interpretado correctamente, pues recibe todo el archivo como texto plano:



Buscando el problema en StackOverflow, descubrimos que el servidor local de Python no puede interpretar archivos php. En la misma consulta, se menciona que php tiene un servidor local también, y que este puede interpretar correctamente los archivos php. (<https://stackoverflow.com/questions/12235876/python-simplehttpserver-with-php?rq=3>)

Ponemos en práctica esta solución.



Ahora, el php se interpreta correctamente.





**Ejemplo 8**

Al ser similar al problema anterior, podemos reutilizar el index.html, modificando solo el archivo “suggestions.php” por “suggestions.asp”. El archivo .asp debería contener la misma lógica que el .php anterior, dando el mimso resultado. (“debería”, porque no fui capaz de encontrar un servidor que pudiese interpretar archivos asp)

<!DOCTYPE html>

<html>

<head>

    <title>Prueba AJAX 8</title>

    <meta charset = "utf-8">

</head>

<body>

    <h1>Prueba AJAX 8</h1>

    <h2>Escriba un nombre debajo:</h2>

    <p>Posibles predicciones: <span id="predictions"></span></p>

    <p>Nombre: <input type="text" id="nameBox" onkeyup=predict(this.value)></p>

    <br>

    <script>

        function predict(str){

            if(str.length == 0){

                document.getElementById("predictions").innerHTML = "Ingrese un nombre";

                return;

            }

            const xhttp = new XMLHttpRequest();

            xhttp.onload = function() {

                console.log(this.response);

                document.getElementById("predictions").innerHTML = this.response;

            }

            xhttp.open("GET", "suggestions.asp?q=" + str, true);

            xhttp.send();

        }

    </script>

</body>

</html>

<%

Dim names

names = Array("Isaac", "Magdalene", "Cain", "Judas", "Blue", "Eve", "Samson", "Azazel", "Lazarus", "Eden", "Lost", "Lilith", "Keeper", "Apollyon", "Forgotten", "Bethany", "Jacob", "Esau")

Dim q

q = Request.QueryString("q")

Dim predict

predict = ""

If Len(q) > 0 Then

    q = LCase(q)

    Dim name

    For Each name In names

        If InStr(LCase(name), q) > 0 Then

            If Len(predict) = 0 Then

                predict = name

            Else

                predict = predict & ", " & name

            End If

        End If

    Next

End If

If Len(predict) = 0 Then

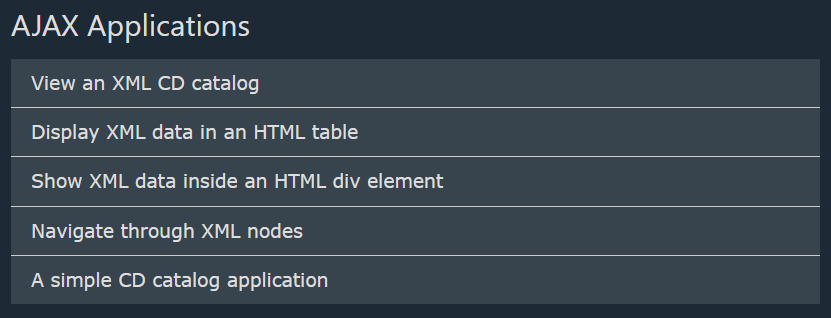
    Response.Write "no hay coincidencias"

Else

    Response.Write predict

End If

%>



Los ejemplos listados aquí son repeticiones de ejemplos pasados (Ejemplo 6), por lo que no es necesario desarrollarlos nuevamente.