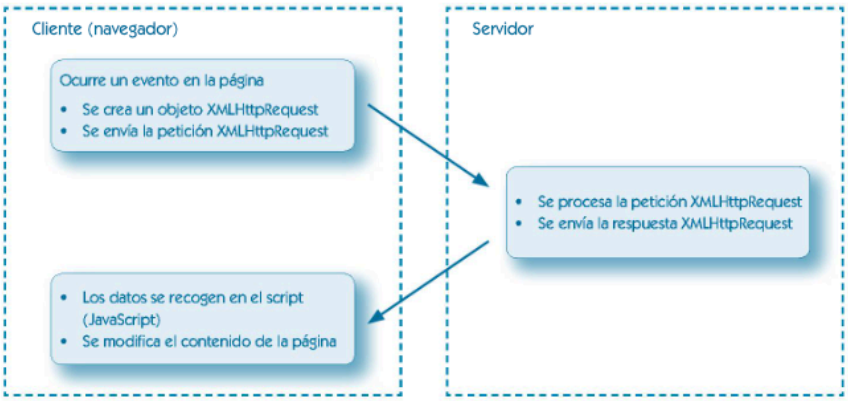
COMUNICACIÓN ASÍNCRONA AJAX.

Con esta técnica se pueden crear páginas dinámicas de forma rápida que consultan de una forma asíncrona fuentes de datos externas a la página. Permite que una página se actualice sin tener que enviar información otra vez al servidor. Se queda esperando a que el servidor le responda y envié su información e vuelta.



El elemento principal de AJAX es el objeto XMLHttpRequest, que permite intercambiar datos con el servidor de forma asíncrona. En concreto el algoritmo de conexión es:

1. Se crea un nuevo objeto **XMLHttpRequest**

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

2. Se llama a la página web donde se van a recibir los datos - **open()**
3. Se envía la petición - **send ()**

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

4. Se va a realizar una petición asíncrona, por lo que se tiene que indicar que código ha de ejecutarse cuando ocurra algo con la petición que se ha realizado, es decir, cada vez que ocurra algo en la petición, se ejecutará la función asociada al evento o al método onreadystatechange del objeto XMLHttpRequest.
5. Una vez que se está dentro de la función, se recoge el valor recibido y se hace con ello lo que proceda (por ejemplo, asignarlo a un elemento HTML). Previamente se evalúa que this.readyState == 4 && this.status == 200, o lo que es lo mismo, comprueba que petición ha terminado completada y que la petición se completado OK (404 es NOT FOUND). Se pueden utilizar las comprobaciones oportunas que se quieran en cada caso.

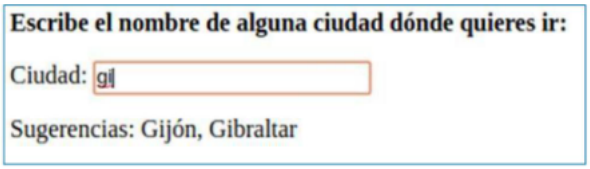
```
xhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
    // Acciones a realizar cuando la solicitud es exitosa
    console.log(this.responseText);
  }
};
```

TOMA NOTA

Valores del atributo readyState:

- 0 (no inicializada).
- 1 (leyendo).
- 2 (leído).
- 3 (interactiva).
- 4 (completo).

Por ejemplo, se tiene el siguiente caso en el que se introduce en una caja de texto un par de letras y el sitio web devuelve sugerencias de nombres de ciudades.




```
<html>
<head>
<script>
function muestraSugerencia(str) {
    if (str.length == 0) {
        document.getElementById("txtSugerencia").innerHTML = "";
        return;
    } else {
        var xmlhttp = new XMLHttpRequest();
        xmlhttp.onreadystatechange = function() {
            if (this.readyState == 4 && this.status == 200) {
                document.getElementById("txtSugerencia").innerHTML = this.
                responseText;
            }
        };
        xmlhttp.open("GET", "http://localhost/Proyectos/JC/getSugerencia.
        php?param=" + str, true);
        // La dirección anterior corresponde a la ubicación de la página
        php en el servidor localhost
        // Esta dirección podrá diferir dependiendo de la necesidad del pro-
        gramador.

            xmlhttp.send();
        }
    }
    </script>
    </head>
    <body>

    <p><b>Escribe el nombre de alguna ciudad d&oacute;nde quieres ir:</b></p>
    <form>
    Ciudad: <input type="text" onkeyup="muestraSugerencia(this.value)">
    </form>
    <p>Sugerencias: <span id="txtSugerencia"></span></p>
    </body>
    </html>
```

GET Y POST AJAX

Las solicitudes AJAX (Asynchronous JavaScript and XML) son un conjunto de técnicas utilizadas en el desarrollo web para enviar y recibir datos de un servidor de forma asíncrona, sin necesidad de recargar toda la página.

GET:

El método GET se utiliza para recuperar datos del servidor. Los datos se envían como parte de la URL (cadena de consulta) después del signo de interrogación (`?`).

Los datos se muestran en la URL, lo que los hace visibles y limitados en términos de cantidad y seguridad.

```
const xhr = new XMLHttpRequest();
xhr.open('GET', 'https://api.example.com/data', true);
xhr.onreadystatechange = function () {
  if (xhr.readyState === 4 && xhr.status === 200) {
    const responseData = JSON.parse(xhr.responseText);
    // Procesar los datos recibidos
  }
};
xhr.send();
```

POST:

El método POST se utiliza para enviar datos al servidor para ser procesados. Los datos se envían en el cuerpo de la solicitud, no en la URL. Puede enviar datos más grandes y sensibles. Los datos no son visibles en la URL, lo que proporciona una capa adicional de seguridad y permite enviar datos más grandes.

```
const xhr = new XMLHttpRequest();
xhr.open('POST', 'https://api.example.com/save', true);
xhr.setRequestHeader('Content-Type', 'application/json');
xhr.onreadystatechange = function () {
  if (xhr.readyState === 4 && xhr.status === 200) {
    const responseData = JSON.parse(xhr.responseText);
    // Procesar los datos recibidos después de guardar en el servidor
  }
};
const dataToSend = { key: 'value' };
xhr.send(JSON.stringify(dataToSend));
```

AJAX Y JSON

Para tratar la información recibida en formato JSON por un script JavaScript al recibir los datos se utilizará el método **parse** para interpretar la información recibida.

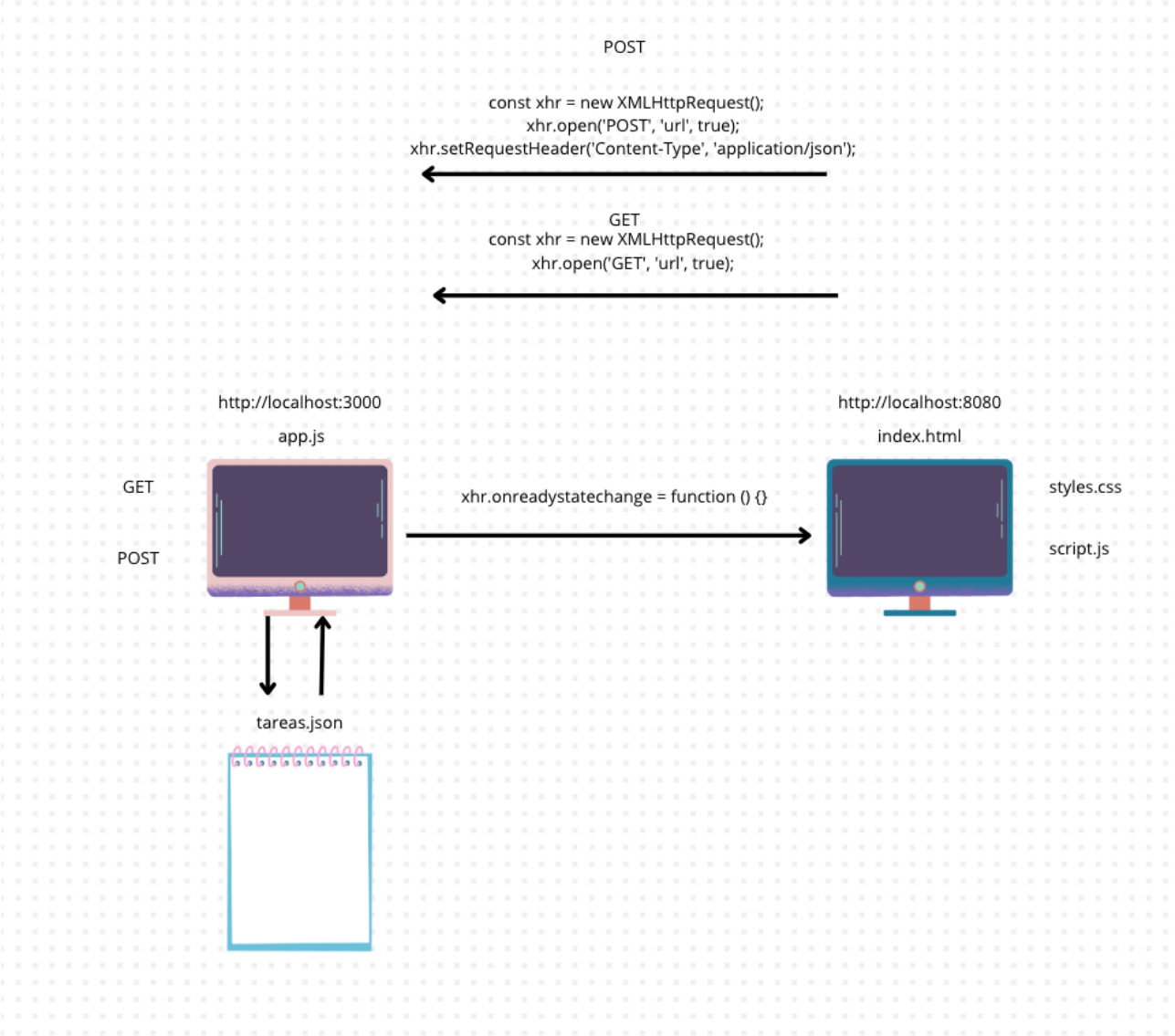
Esto crea un array de objetos que se puede analizar y trabajar sobre él de la forma habitual

```
<html>
<head>
</head>
<body>
<div id="lasWebs"></div>
<script>
       var xmlhttp = new XMLHttpRequest();
       var url = "mywebs.txt";
       xmlhttp.onreadystatechange = function() {
       if (this.readyState == 4 && this.status == 200) {
           var listaWebs = JSON.parse(this.responseText);
           getWebs(listaWebs);
       }
       };
       xmlhttp.open("GET", url, true);
       xmlhttp.send();

      function getWebs(arr) {
      var out = "";
      var i;
      for(i = 0; i < arr.length; i++) {
          out += '<a href="' + arr[i].url + '">' +
          arr[i].display + '</a><br>';
      }
      document.getElementById("lasWebs").innerHTML = out;
      }
</script>
</body>
</html>
```

Ejercicio. Implementa el sitio web de consulta de ciudades que previamente se han cargado en fichero json.

ANEXO 1. PROYECTO CON APP.JS (SERVIDOR CON EXPRESS) Y AJAX



ANEXO 2. ¿Cómo quedaría el archivo en PHP al que se le hace la petición?

```php
<?php
// Array de nombres de ciudades
$arr[] = "Casares";
$arr[] = "Barcelona";
$arr[] = "Alcorcón";
$arr[] = "Málaga";
$arr[] = "Fuengirola";
$arr[] = "Marbella";
$arr[] = "Río de Janeiro";
$arr[] = "Gijón";
$arr[] = "Oviedo";
$arr[] = "Cáceres";
$arr[] = "Mérida";
$arr[] = "Estepona";
$arr[] = "París";
$arr[] = "Londres";
$arr[] = "Berlín";
$arr[] = "Moscú";
$arr[] = "Atenas";
$arr[] = "Kassel";
$arr[] = "Praga";
$arr[] = "Nueva York";
$arr[] = "Dublín";
$arr[] = "Manilva";
$arr[] = "Gibraltar";
$arr[] = "Ulan Bator";
$arr[] = "Bombay";
$arr[] = "Niza";
$arr[] = "Pekín";
$arr[] = "Tokio";
$arr[] = "Vasteras";
$arr[] = "Washington";
$arr[] = "México DC";
$arr[] = "Estocolmo";
$arr[] = "Stocholm";

// Primero recogemos el parámetro de la URL
$param = $_REQUEST["param"];

$sugerencia = "";

// Mira si coincide el parámetro con alguna de las ciudades
if ($param !== "") {
      $param = strtolower($param); //convierte el parámetro recibido a
      minúscula

       $len=strlen($param); //determinamos la longitud del parámetro re-
       cibido
          foreach($arr as $nombre) {
       // comparamos si el texto recibido coincide con el comienzo de al-
       guna ciudad registrada en nuestro array
             if (stristr($param, substr($nombre, 0, $len))) {
                if ($sugerencia === "") {
             $sugerencia = $nombre;  // primera sugerencia
                } else {
             $sugerencia .= ", $nombre"; //segunda y siguientes sugerencias
                }
             }
          }
}
// En el caso de no encontrar ninguna sugerencia se le hace saber al usuario.
// en caso contrario devuelve las sugerencias encontradas
echo $sugerencia === "" ? "ninguna sugerencia" : $sugerencia;
?>
```