**Actividad mostrar tabla con datos locales (sin uso de base de datos).**

**Requisitos**

* Tener instalado el servidor Apache, este ejemplo se probó con WAMP server 3.0.6 de 64 bits.
* Tener instalado un editor de código, en estos ejemplos se trabajó con Microsoft Visual Studio Code 1.27.2.
* Conocimiento básico de PHP.
* Conocimiento básico de html y tablas de html.
* Conocimiento básico de JS.

**1. Antecedentes.**

**2 Desarrollo.**

***2.1. Crear ruta***

Como sabe los archivos PHP deben almacenarse en la carpeta www de la carpeta WAMP, en mi caso: “C:\wamp64\www”, si instalo la versión de 32 bits la carpeta será: “C:\wamp\www”, para trabajar de manera organizada, en esta carpeta crearemos una carpeta nueva: “actividades”.

En la carpeta nueva creamos la subcarpeta “json-local-crearTabla”

La ruta debe quedar así: “C:\wamp64\www\actividades\json-local-crearTabla”

***2.2. Cargar archivos***

En la carpeta “json-local-crearTabla” compartida por el profesor encontrara los 2 archivos necesarios para la práctica. Cópielos a la ruta: “C:\wamp64\www\actividades\json-local-crearTabla”

***2.3. codificar PHP***

Escriba el código en PHP para almacenar los datos:

| **id** | **nombre** | **edad** | **ciudad** | **trabajo** |
| --- | --- | --- | --- | --- |
| 1 | pedro | 30 | Cármen | electronico |
| 2 | josé | 20 | Cármen | carpintero |
| 3 | alma maria | 10 | Campeche | albañil |
| 4 | rosa | 50 | Campeche | plomero |
| 5 | agapito | 23 | Cármen | mecanico |
| 6 | isidro | 38 | Merida | deportista |
| 7 | roman | 42 | Cármen | panadero |
| 8 | raul | 41 | Mérida | electronico |

Tabla 1datos de personas

Un arreglo asociativo en PHP se crea así:

|  |
| --- |
|  |
| $datos=[  0 => [  'id' => '1' ,  'nombre' => 'pedro' ,  'apellido' => 'nuño' ,  'edad' => '30',  'ciudad' => 'Cármen' ,  'trabajo' => 'electronico' ],  1 => [  'id' => '2' ,  'nombre' => 'josé' ,  'apellido' => 'pérez' ,  'edad' => '20',  'ciudad' => 'Cármen' ,  'trabajo' => 'carpintero' ]  ]; |

Tabla 2 Datos generados desde php

Complete el código para almacenar todos los datos y devolverlos al cliente.

***2.4 Solución***

En su editor de código abra el archivo: “devuelve-json-local.php” que encontrará en los archivos copiados en el punto 2.2, ahí encontrará el código PHP que es la solución a esta parte.

Comente con el profesor sus dudas.

***2.5 Probar el archivo***

***Arranque su servidor.***

Si instalo el WAMP, en el escritorio debe ver el siguiente icono:



De doble clic sobre él, para arrancar el servidor, espere unos instantes hasta que en la barra inferior del lado derecho aparezca el símbolo:

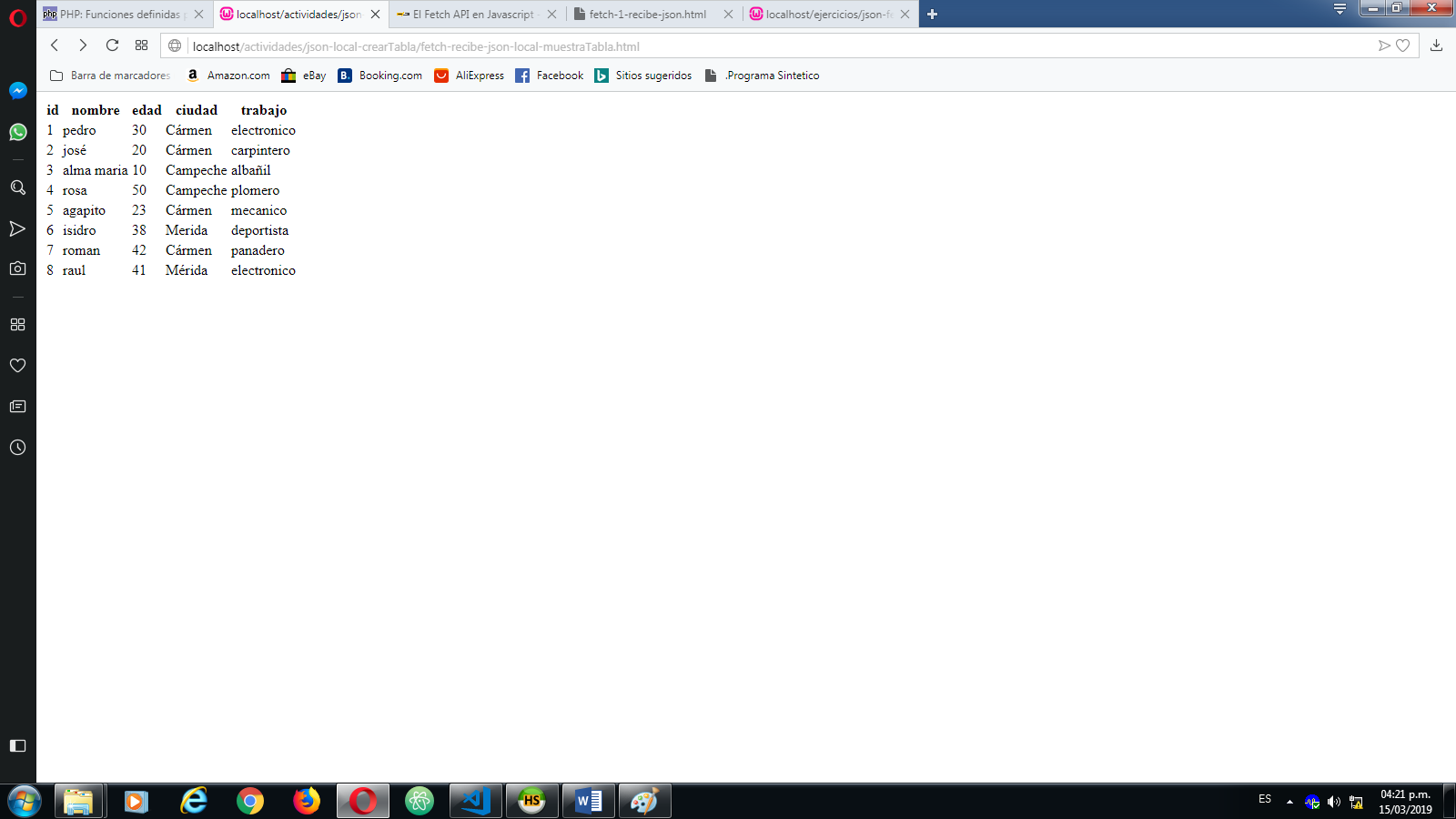


Que indica que el servidor está corriendo.

Pruebe el archivo en su navegador de esta forma: Escriba en la ruta de su navegador la siguiente dirección:

<http://localhost/actividades/json-local-crearTabla/fetch-recibe-json-local-muestraTabla.html>

Debe ver lo siguiente:



***2.6 Leer datos desde JS.***

Escriba el código para leer datos de un archivo PHP, tome esos datos y cárguelos en una tabla. En la liga: <https://developer.mozilla.org/es/docs/Web/API/Fetch_API/Utilizando_Fetch> puede ver una referencia a la documentación de la API fetch.

***2.7 Solución.***

En su editor de código abra el archivo: “fetch-recibe-json-local-muestraTabla.html” que encontrará en los archivos copiados en el punto 2.2, ahí encontrará el código html/js que es la solución a esta parte.

Explicación.

En este archivo encuentra la definición de una tabla con tbody identificador “datosTabla”

|  |
| --- |
| <table>  <thead>  <tr>  <th>id</th>  <th>nombre</th>  <th>edad</th>  <th>ciudad</th>  <th>trabajo</th>  </tr>  </thead>  <tbody id="datosTabla">  </tbody>  </table> |

Tabla 3 Código html

El código js es el siguiente:

|  |
| --- |
| var personas={};  fetch('devuelve-json-local.php')  .then(function(response) {  return response.json();  })  .then(function(data) {  //console.log('data = ', data);  let personas=data;  var content="";  personas.forEach(function(element) {  content+="<tr>";  //nota: como se ve a continuación,  //se accede al json con: element["id"] o con:element.id  content+="<td>"+element["id"]+"</td><td>"+element.nombre+"</td>";  content+="<td>"+element["edad"]+"</td><td>"+element['ciudad']+"</td>";  content+="<td>"+element.trabajo;  content+="</tr>";  });  document.getElementById("datosTabla").innerHTML=content;  })  .catch(function(err) {  console.error(err);  }); |

Tabla 4 Código js para generar datos de tabla

Para entender este código vemos primero que son las promesas.

***2.8 Promesas.***

Una promesa es un objeto que representa el éxito o fracaso de una operación asíncrona. Una operación asíncrona es una operación que no se ejecuta una tras otra, por ejemplo, si ejecuta de manera síncrona las sentencias:

Sentencia 1;

Sentencia 2;

Sentencia 3

La sentencia 2 no se ejecuta hasta que se concluya la sentencia 2, y la sentencia 3 se ejecuta hasta que se concluya la sentencia 2.

En la ejecución asíncrona no sucede así.

En la siguientes 2 línea de pseudo código se muestra una operación asíncrona típica:

1. fetch('http://example.com/movies.php”) //solicitar datos al servidor
2. Hacer algo con el resultado

Las solicitudes a fetch se lanzan de manera asíncrona, esto quiere decir que se lanza la solicitud fetch y la ejecución continua sin esperar que la operación fetch concluya, esto crea el problema siguiente:

Como el paso 1 se lanza de manera asíncrona, quiere decir que se lanza la petición de datos y sin esperar la respuesta (por que la red puede estar lenta, por ejemplo) el navegador se ira al paso 2, pero esto puede generar un error porque los datos tal vez aun no estén listos, para esto se escriben funciones encadenadas.

Las operaciones encadenadas se usan cuando queremos ejecutar operaciones asíncronas donde cada operación posterior queremos que se inicie cuando la operación anterior se haya terminado. Veamos el uso de encadenamiento utilizando la sentencia then.

hazAlgo().then(function(resultado) {

return hazAlgoMas(resultado);

})

.then(function(nuevoResultado) {

return hazLaTerceraCosa(nuevoResultado);

})

.then(function(resultadoFinal) {

console.log('Obtenido el resultado final: ' + resultadoFinal);

})

.catch(falloCallback);

Conjunto de funciones encadenadas, donde cada una se ejecuta cuando se obtiene el resultado de la anterior.

Para nuestro caso en el uso del fetch tenemos:

fetch('devuelve-json-local.php')

.then(function(response) {

return response.json();

})

.then(function(myJson) {

console.log(myJson);

});

Funciona así: la función fetch permite acceder al http para solicitar y recibir datos de un servidor. En este caso estamos indicando que queremos ejecutar en el servidor local el archivo 'devuelve-json-local.php', que como vimos antes devuelve un conjunto de datos. A continuación del fetch usamos

.then (function(response) {

return response.json();

})

indica la función a ejecutar cuando se reciba la respuesta del fetch. Además, la respuesta del fetch se recibe como un parámetro, en este caso llamado “response”, esta función observe que la única sentencia que tiene es:

retrun response.json();

Esta sentencia toma el parámetro de entrada (que el conjunto de datos enviados por el fetch), lo convierte a formato json y lo devuelve.

Esta segunda función tiene solo una sentencia:

return response.json();

que convierte su entrada response al formato json y la devuelve, como hay una segunda función encadenada, esta segunda función recibe la respuesta de la primera función encadenada, es decir los datos en formato json y los muestra en la consola.

Ahora ya podemos entender el funcionamiento del código JS de la tabla4. Funciona así:

Fecth(“archivophp”) //solicita datos

.then (function(resultadodelFetch){ //toma el resultado del fetch

return resultadodelFetch.json(); //conviente el resultado recibido al formato json

})

.then(data){ //recibe los datos en formato json

//puebla la tabla con los datos recibidos

}

***2.9 Solución.***

Probar el código: pruebe el código escribiendo esto en su navegador: <http://localhost/actividades/json-local-crearTabla/fetch-recibe-json-local-muestraTabla.html>

Nota. Estudie el código de esta práctica y comente con el profesor.