

DESARROLLO WEB EN ENTORNO CLIENTE

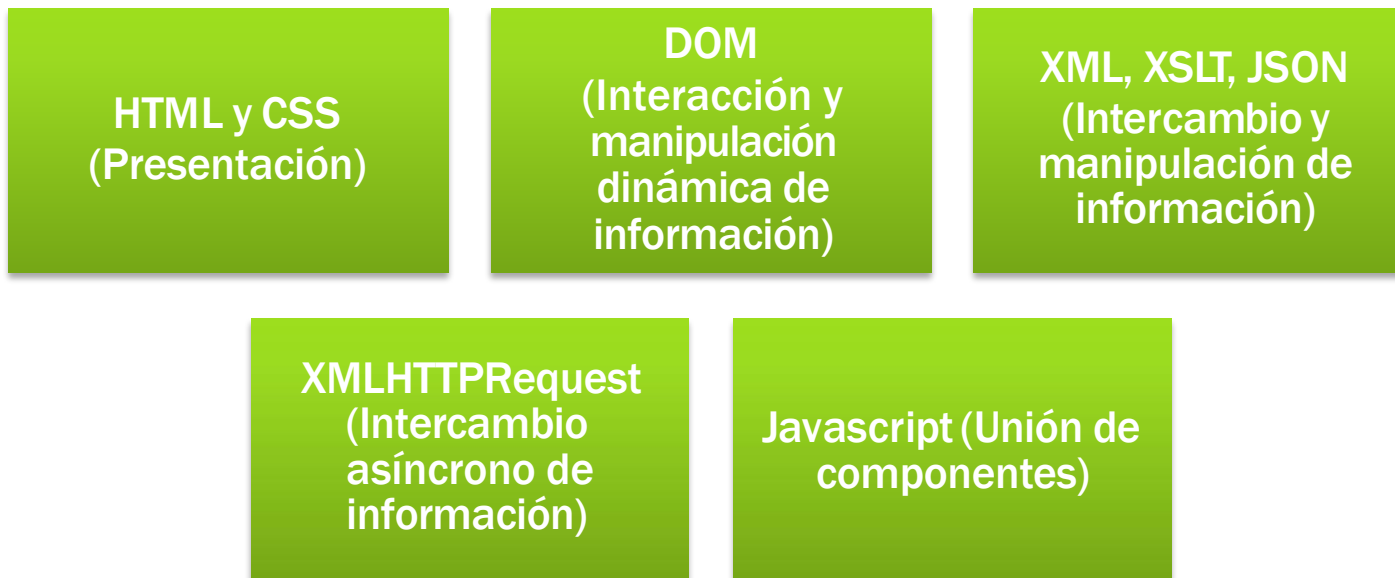
7. UTILIZACIÓN DE MECANISMOS DE COMUNICACIÓN
ASÍNCRONA

9. UTILIZACIÓN DE MECANISMOS DE COMUNICACIÓN ASÍNCRONA

1. INTRODUCCIÓN A AJAX.

1. AJAX

Elementos de Ajax



9. UTILIZACIÓN DE MECANISMOS DE COMUNICACIÓN ASÍNCRONA

1. INTRODUCCIÓN A AJAX.

1. AJAX

SIN AJAX

El usuario hace solicitud desde la interfaz.

Se hace una solicitud HTTP al servidor.

El cliente tiene que esperar hasta recibir la respuesta (cambio de página)

El servidor realiza la acción.

El servidor devuelve la página al cliente.

CON AJAX

El usuario hace solicitud desde la interfaz.

Se hace una solicitud HTTP al servidor.

El cliente sigue navegando por la página.

Cuando el servidor realiza la acción muestra al cliente sin recargar la página.



9. UTILIZACIÓN DE MECANISMOS DE COMUNICACIÓN ASÍNCRONA

1. INTRODUCCIÓN A AJAX.

1. AJAX



9. UTILIZACIÓN DE MECANISMOS DE COMUNICACIÓN ASÍNCRONA

1. INTRODUCCIÓN A AJAX.

1. AJAX

Requisitos

Servidor web

- Apache, IIS...

Servidor de bases de datos

- MySQL, Postgresql...

Lenguaje de servidor

- PHP, ASP...

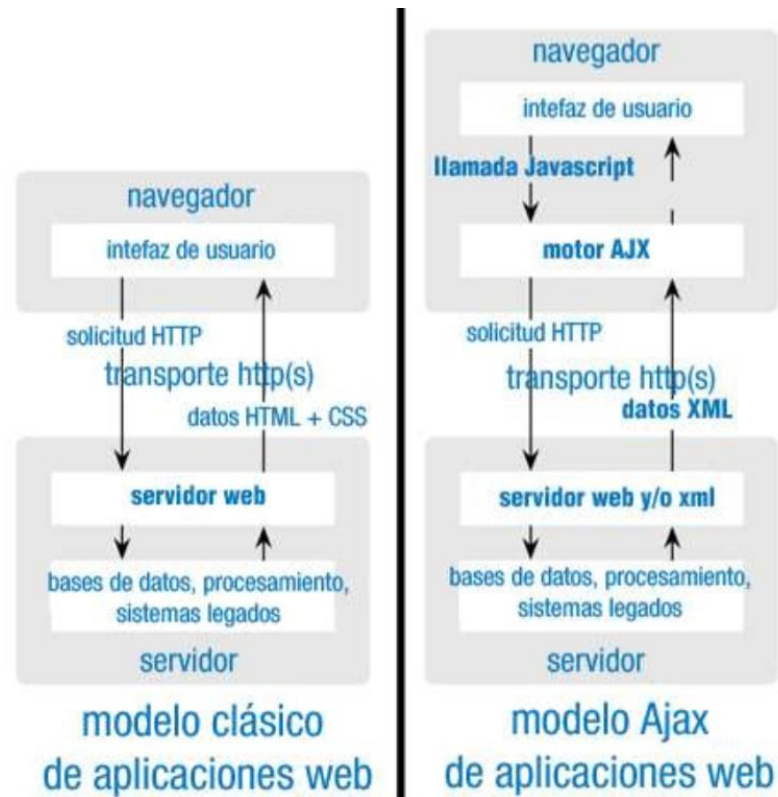


9. UTILIZACIÓN DE MECANISMOS DE COMUNICACIÓN ASÍNCRONA

9.1. INTRODUCCIÓN A AJAX.

9.1.2. COMUNICACIÓN ASÍNCRONA

Comunicación asíncrona



9. UTILIZACIÓN DE MECANISMOS DE COMUNICACIÓN ASÍNCRONA

9.1. INTRODUCCIÓN A AJAX.

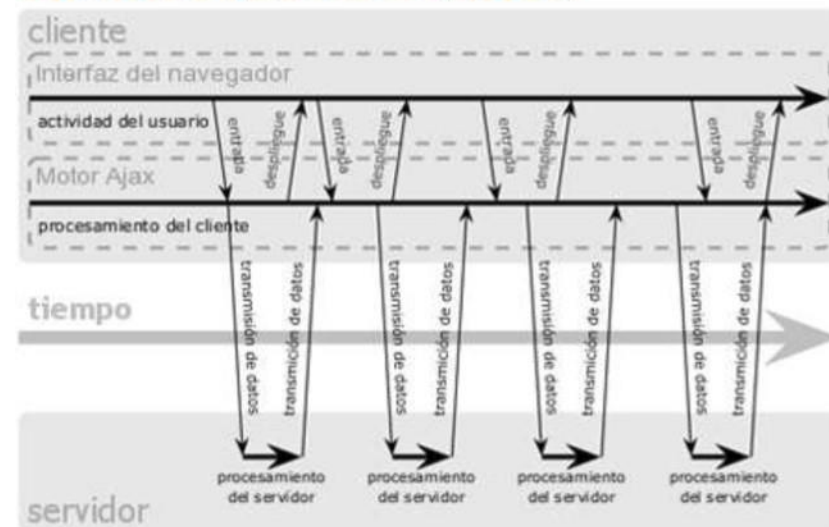
9.1.2. COMUNICACIÓN ASÍNCRONA

Comunicación asíncrona

modelo clásico de aplicaciones web (síncrono)



modelo Ajax de aplicaciones web (asíncrono)



9. UTILIZACIÓN DE MECANISMOS DE COMUNICACIÓN ASÍNCRONA

9.1. INTRODUCCIÓN A AJAX.

9.1.3. EL API XMLHttpRequest (XHR)

XMLHttpRequest (XHR):

- API de lenguajes de script del lado del cliente (ej. Javascript).
- Se utiliza en peticiones al servidor web (http o https).
- Los datos se pueden recibir en texto plano o XML y modificar el DOM sin modificar la página.
- Los datos también se pueden recibir en formato JSON y ser evaluados con Javascript.
- No es posible modificar páginas alojadas fuera del dominio desde el que se realiza la petición.



9. UTILIZACIÓN DE MECANISMOS DE COMUNICACIÓN ASÍNCRONA

9.1. INTRODUCCIÓN A AJAX.

9.1.3. EL API XMLHttpRequest (XHR)

Propiedades del objeto XMLHttpRequest (XHR):

Propiedad	Descripción
readyState	Valor numérico (entero) que almacena el estado de la petición
responseText	El contenido de la respuesta del servidor en forma de cadena de texto
responseXML	El contenido de la respuesta del servidor en formato XML. El objeto devuelto se puede procesar como un objeto DOM
status	El código de estado HTTP devuelto por el servidor (200 para una respuesta correcta, 404 para "No encontrado", 500 para un error de servidor, etc.)
statusText	El código de estado HTTP devuelto por el servidor en forma de cadena de texto: "OK", "Not Found", "Internal Server Error", etc.



9. UTILIZACIÓN DE MECANISMOS DE COMUNICACIÓN ASÍNCRONA

9.1. INTRODUCCIÓN A AJAX.

9.1.3. EL API XMLHttpRequest (XHR)

Valores de la propiedad `readyState`:

Valor	Descripción
0	No inicializado (objeto creado, pero no se ha invocado el método <code>open</code>)
1	Cargando (objeto creado, pero no se ha invocado el método <code>send</code>)
2	Cargado (se ha invocado el método <code>send</code> , pero el servidor aún no ha respondido)
3	Interactivo (se han recibido algunos datos, aunque no se puede emplear la propiedad <code>responseText</code>)
4	Completo (se han recibido todos los datos de la respuesta del servidor)



9. UTILIZACIÓN DE MECANISMOS DE COMUNICACIÓN ASÍNCRONA

9.1. INTRODUCCIÓN A AJAX.

9.1.3. EL API XMLHttpRequest (XHR)

Métodos del objeto XMLHttpRequest (XHR):

Método	Descripción
abort()	Detiene la petición actual
getAllResponseHeaders()	Devuelve una cadena de texto con todas las cabeceras de la respuesta del servidor
getResponseHeader("cabecera")	Devuelve una cadena de texto con el contenido de la cabecera solicitada
onreadystatechange	Responsable de manejar los eventos que se producen. Se invoca cada vez que se produce un cambio en el estado de la petición HTTP. Normalmente es una referencia a una función JavaScript



9. UTILIZACIÓN DE MECANISMOS DE COMUNICACIÓN ASÍNCRONA

9.1. INTRODUCCIÓN A AJAX.

9.1.3. EL API XMLHttpRequest (XHR)

Métodos del objeto XMLHttpRequest (XHR):

Método	Descripción
<code>open("metodo", "url")</code>	Establece los parámetros de la petición que se realiza al servidor. Los parámetros necesarios son el método HTTP empleado y la URL destino (puede indicarse de forma absoluta o relativa)
<code>send(contenido)</code>	Realiza la petición HTTP al servidor
<code>setRequestHeader("cabecera", "valor")</code>	Permite establecer cabeceras personalizadas en la petición HTTP. Se debe invocar el método <code>open()</code> antes que <code>setRequestHeader()</code>



9. UTILIZACIÓN DE MECANISMOS DE COMUNICACIÓN ASÍNCRONA

9.1. INTRODUCCIÓN A AJAX.

9.1.3. EL API XMLHttpRequest (XHR)

Métodos del objeto XMLHttpRequest (XHR):

El método open:

- `open(string metodo, string URL [,boolean asincrono, string usuario, string password]);`
- Los dos primeros parámetros son obligatorios, los otros tres opcionales.
- Si se indica en el tercer parámetro “false”, las peticiones se hacen síncronas.
- Realizar peticiones síncronas es contrario a la filosofía de Ajax.

El método send:

- En el parámetro se indica la información a enviar al servidor.
- Si se envían datos, puede ser una cadena, un array de bytes o un XML. Si no se envían datos, debe ser un valor null.



9. UTILIZACIÓN DE MECANISMOS DE COMUNICACIÓN ASÍNCRONA

9.1. INTRODUCCIÓN A AJAX.

9.1.3. EL API XMLHttpRequest (XHR)

Programa de ejemplo: Hola mundo.

```
<!DOCTYPE html>
<html>

<head>

<title>Hola Mundo con AJAX</title>

</head>
<body>
  <div id="texto">
    <h1>AJAX</h1>
    <button id="cambiaContenido">Cambia el contenido</button>
  </div>

  <script>
    document.getElementById("cambiaContenido").addEventListener("click",
    cambiaContenido);

    function cambiaContenido() {

      // Obtener la instancia del objeto XMLHttpRequest
      let peticion_http = new XMLHttpRequest();

      // Preparar la funcion de respuesta
      peticion_http.onreadystatechange = muestraContenido;

      // Realizar peticion HTTP. Get sin envío de parámetros
      /* .open: especifica la solicitud
      - GET/POST.
      - Archivo: txt, php, xml, json, etc.
      - true/false: método de envío. */
      peticion_http.open("GET","holamundo.txt",true);
```

```
/* .send: envía la solicitud al servidor.
Si utilizamos POST debemos pasar los datos por parámetro */
peticion_http.send();

function muestraContenido() {
  //Ha recibido una respuesta y la respuesta es válida y correcta
  if (this.readyState == 4 && this.status == 200) {
    document.getElementById("texto").innerHTML = this.responseText;
  }
}

</script>

</body>

</html>
```



9. UTILIZACIÓN DE MECANISMOS DE COMUNICACIÓN ASÍNCRONA

9.1. INTRODUCCIÓN A AJAX.

9.1.3. EL API XMLHTTPREQUEST (XHR)

Programa de ejemplo: Hola mundo (mejorado)

```
<!DOCTYPE html>
<html>
<head>
<title>Hola Mundo con AJAX, version 2</title>
</head>
<body>
  <div id="texto">
    <h1>AJAX</h1>
    <button id="cambiaContenido">Cambia el contenido</button>
  </div>
<script>
  let READY_STATE_UNINITIALIZED=0;
  let READY_STATE_LOADING=1;
  let READY_STATE_LOADED=2;
  let READY_STATE_INTERACTIVE=3;
  let READY_STATE_COMPLETE=4;
```

```
document.getElementById("cambiaContenido").addEventListener("click", cambiaContenido);

function cambiaContenido() {
  // Obtener la instancia del objeto XMLHttpRequest
  let peticion_http = new XMLHttpRequest();

  // Preparar la funcion de respuesta
  if (peticion_http){
    peticion_http.onreadystatechange = muestraContenido;
    peticion_http.open("GET","holamundo.txt",true);
    peticion_http.send();
  }

  function muestraContenido() {
    //Ha recibido una respuesta y la respuesta es válida y correcta
    if (this.readyState == READY_STATE_COMPLETE && this.status == 200) {
      document.getElementById("texto").innerHTML = this.responseText;
    }
  }
</script>
</body>
</html>
```



9. UTILIZACIÓN DE MECANISMOS DE COMUNICACIÓN ASÍNCRONA

9.1. INTRODUCCIÓN A AJAX.

9.1.3. EL API XMLHttpRequest (XHR)

Programa de ejemplo: solicitud fecha al servidor (index.html)

```
<!DOCTYPE html>
<head>
  <meta charset="UTF-9">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Ejemplo dwec09 - 2.3 - AJAX ASINCRONO GET - POST</title>
  <script src="index.js"></script >
  <style> #resultados{ background: yellow; } </style>
</head>
<body>
  A continuación se cargarán por AJAX los datos recibidos en la solicitud ASINCRONA:
  <br/>
  Contenedor resultados:
  <div id="resultados"></div>
  <div id="indicador"></div>
</body>
</html>
```



9. UTILIZACIÓN DE MECANISMOS DE COMUNICACIÓN ASÍNCRONA

9.1. INTRODUCCIÓN A AJAX.

9.1.3. EL API XMLHttpRequest (XHR)

Programa de ejemplo: solicitud fecha al servidor
(index.js)

```
//Cuando se carga el documento comienza iniciar()

crearEvento(window,"load",iniciar);

function iniciar()
{
    // Creamos un objeto XHR.
    miXHR = new XMLHttpRequest();

    // Cargamos el fichero fecha.php de forma asíncrona.
    cargarAsync("fecha.php");
}
```

```
// Carga el contenido de la url de forma asíncrona con Ajax
function cargarAsync(url)
{
    if (miXHR) {
        // Carga el indicador Ajax antes de realizar la petición.
        document.getElementById("indicador").innerHTML=" <img src='ajax-loader.gif'/>";
        //Si existe el objeto miXHR abrimos la url (asíncrona)
        miXHR.open('GET', url, true);
        // En cada cambio de estado llama a estadoPetición
        miXHR.onreadystatechange = estadoPetición;
        // Hacemos la petición al servidor con GET y parámetro null
        miXHR.send();
    }
}
```



9. UTILIZACIÓN DE MECANISMOS DE COMUNICACIÓN ASÍNCRONA

9.1. INTRODUCCIÓN A AJAX.

9.1.3. EL API XMLHTTPREQUEST (XHR)

Programa de ejemplo: solicitud fecha al servidor (index.js)

```
// Se llama en cada cambio de estado de la petición.

// 1. readyState == 4 cuando la petición ha terminado.
// 2. Status == 200 encontrado; ==404 no encontrado...

function estadoPetición(){
    if ( this.readyState==4 && this.status == 200 ){
        // Desactivamos el indicador AJAX.
        document.getElementById("indicador").innerHTML="";
        // Metemos en el contenedor resultados las respuestas de la petición AJAX.
        textoDIV(document.getElementById("resultados"), this.responseText);
    }
}
```

Programa de ejemplo: solicitud fecha al servidor (fecha.php)

```
<?php

// Para que el navegador no haga cache de los datos devueltos por la página
// PHP.
header('Cache-Control: no-cache, must-revalidate');

header('Expires: Mon, 26 Jul 1999 05:00:00 GMT');
// retrasamos 2 segundos la ejecución de esta página PHP.
sleep(2);
// Mostramos la fecha y hora del servidor web.
echo "La fecha y hora del Servidor Web: "; echo date("j/n/Y G:i:s.");

?>
```



9. UTILIZACIÓN DE MECANISMOS DE COMUNICACIÓN ASÍNCRONA

9.1. INTRODUCCIÓN A AJAX.

9.1.3. EL API XMLHttpRequest (XHR)

Programa de ejemplo: solicitud nombre y apellidos con GET (funciones iniciar y cargar de index.js)

```
function iniciar() {  
  
    // Creamos un objeto XHR  
  
    miXHR = new XMLHttpRequest();  
  
    // Cargamos de forma asíncrona el texto que nos devuelve la página procesar.php con los  
    // parámetros indicados en la URL  
  
    cargarAsync("procesar.php?nombre=James&apellido=Bond");  
  
}  
//En la función cargarAsync (url) tenemos:  
miXHR.open('GET', url, true); //Abrimos la url, true=ASINCRONA  
  
miXHR.onreadystatechange = estadoPetición  
  
miXHR.send();
```

Programa de ejemplo: solicitud nombre y apellidos con POST (funciones iniciar y cargar async de index.js)

```
function iniciar() {  
  
    // Creamos un objeto XHR  
  
    miXHR = new XMLHttpRequest();  
  
    // Cargamos de forma asíncrona lo que devuelve la página procesar.php  
    // En este caso sólo ponemos los parámetros que pasaremos a la página  
  
    cargarAsync("nombre=James&apellido=Bond");  
  
}  
//En la función cargarAsync (parametros) tenemos:  
miXHR.open("POST", "procesar.php", true); // Abrimos la url, true=ASINCRONA  
  
miXHR.onreadystatechange = estadoPetición  
  
miXHR.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");  
  
miXHR.send(parametros);
```



9. UTILIZACIÓN DE MECANISMOS DE COMUNICACIÓN ASÍNCRONA

9.1. INTRODUCCIÓN A AJAX.

9.1.3. EL API XMLHTTPREQUEST (XHR)

Programa de ejemplo: solicitud nombre y apellidos con GET o POST (procesar.php)

```
<?php

// Para que el navegador no haga cache de los datos devueltos por la página PHP.

header('Cache-Control: no-cache, must-revalidate');

header('Expires: Mon, 26 Jul 1999 05:00:00 GMT');

// Imprimimos un mensaje con los textos recibidos

if (isset($_GET['nombre']))

echo "Saludos desde el servidor: hola {$_GET['nombre']} {$_GET['apellidos']}. ";

else if (isset($_POST['nombre']))

echo "Saludos desde el servidor: hola {$_POST['nombre']} {$_POST['apellidos']}. ";

?>
```



9. UTILIZACIÓN DE MECANISMOS DE COMUNICACIÓN ASÍNCRONA

9.2. FORMATOS PARA EL ENVÍO Y RECEPCIÓN DE LA INFORMACIÓN

Recepción de datos en formato XML

- En la función `iniciar()` solicitamos que cargue, de manera asíncrona, el fichero PHP que devuelve los datos XML. Para ello usamos:
 - `cargarAsync("fichero.php");`
- Al recibir datos en formato XML debemos hacer la solicitud con la propiedad `.responseXML` del objeto XHR.
 - `resultados = this.responseXML;`
- En la función `estadoPetición` realizamos el procesamiento del XML.
 - `Elementos = resultados.documentElement.getElementsByTagName("etiqueta_elementos");`
 - `For (i=0; i<elementos.length; i++)`
- En caso de que haya elementos que puedan estar vacíos, debemos utilizar `try- catch` para evitar excepciones en Javascript.



9. UTILIZACIÓN DE MECANISMOS DE COMUNICACIÓN ASÍNCRONA

9.2. FORMATOS PARA EL ENVÍO Y RECEPCIÓN DE LA INFORMACIÓN

JSON



Formato para el intercambio de datos.

Alternativa a XML.

Fácil uso en Javascript.

Puede ser leído por cualquier lenguaje de programación

9. UTILIZACIÓN DE MECANISMOS DE COMUNICACIÓN ASÍNCRONA

9.2. FORMATOS PARA EL ENVÍO Y RECEPCIÓN DE LA INFORMACIÓN

Sintaxis de JSON

- `"Nombre": "James"`

Valores

- Número (entero o float)
- String (entre comillas simples)
- Booleano (true o false)
- Array (entre corchetes)
- Objeto (entre llaves)
- Null

Objetos

- `{ "NombreFruta": "Manzana", "Cantidad": 20 }`

Arrays

- `{`
- `"Frutas": [`
- `{ "NombreFruta": "Manzana", "cantidad": 10 },`
- `{ "NombreFruta": "Pera", "cantidad": 20 },`
- `{ "NombreFruta": "Naranja", "cantidad": 30 }`
- `]`
- `}`

9. UTILIZACIÓN DE MECANISMOS DE COMUNICACIÓN ASÍNCRONA

9.2. FORMATOS PARA EL ENVÍO Y RECEPCIÓN DE LA INFORMACIÓN

Ejemplo completo de una frutería

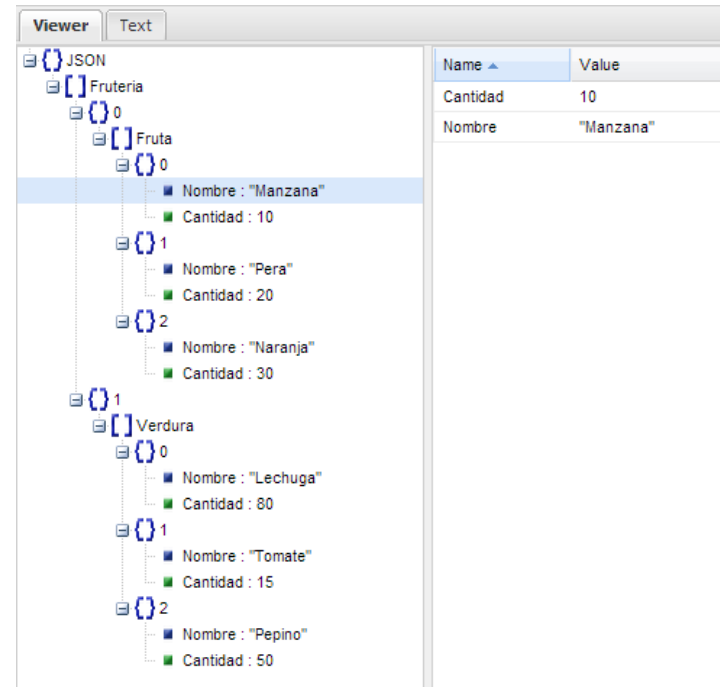
```
{"Fruteria":  
  [  
    {"Fruta":  
      [  
        {"Nombre":"Manzana","Cantidad":10},  
        {"Nombre":"Pera","Cantidad":20},  
        {"Nombre":"Naranja","Cantidad":30}  
      ]  
    },  
    {"Verdura":  
      [  
        {"Nombre":"Lechuga","Cantidad":90},  
        {"Nombre":"Tomate","Cantidad":15},  
        {"Nombre":"Pepino","Cantidad":50}  
      ]  
    }  
  ]  
}
```



9. UTILIZACIÓN DE MECANISMOS DE COMUNICACIÓN ASÍNCRONA

9.2. FORMATOS PARA EL ENVÍO Y RECEPCIÓN DE LA INFORMACIÓN

JSON Viewer



The JSON Viewer application displays a hierarchical tree structure of JSON data. The tree is expanded to show the following structure:

- JSON
 - Fruteria
 - 0
 - Fruta
 - 0
 - Nombre : "Manzana"
 - Cantidad : 10
 - 1
 - Nombre : "Pera"
 - Cantidad : 20
 - 2
 - Nombre : "Naranja"
 - Cantidad : 30
 - 1
 - Verdura
 - 0
 - Nombre : "Lechuga"
 - Cantidad : 80
 - 1
 - Nombre : "Tomate"
 - Cantidad : 15
 - 2
 - Nombre : "Pepino"
 - Cantidad : 50

The right pane shows a table with the following data:

Name	Value
Cantidad	10
Nombre	"Manzana"

9. UTILIZACIÓN DE MECANISMOS DE COMUNICACIÓN ASÍNCRONA

9.2. FORMATOS PARA EL ENVÍO Y RECEPCIÓN DE LA INFORMACIÓN

Recepción de datos en formato JSON

- En la función `iniciar()` solicitamos que cargue, de manera asíncrona, el fichero PHP que devuelve los datos XML. Para ello usamos:
 - `cargarAsync("fichero.php");`
- Al recibir datos en formato JSON debemos hacer la solicitud con la propiedad `.responseText` del objeto XHR y la parseamos con `JSON.parse` teniendo en cuenta que devolverá un array.
 - `resultados = JSON.parse(this.responseText);`
- En la función `estadoPetición` realizamos el procesamiento del JSON. Para el ejemplo de las frutas:
 - `texto = "<table border=1><tr><th>Fruta</th><th>Cantidad</th></tr>";`
 - `var frutas = mijson[0].Fruta;`
 - `for (var i=0; i < frutas.length; i++)`
 - `{`
 - `fruta = frutas[i];`
 - `texto+="`
 - `}`
 - `texto+="`
- En caso de que haya elementos que puedan estar vacíos, debemos utilizar try-catch para evitar excepciones en Javascript.

