

# **Unidad N° 6**

---

## Introducción a la Web 2.0

# Introducción a la Web 2.0

---

## *Introducción*

- Internet y la WWW permitieron a los programadores comenzar a desarrollar aplicaciones web:
  - Aplicación software codificada en un lenguaje soportado por los navegadores web, y que se ejecuta en este último.
- Son populares debido a lo práctico del navegador web como cliente, así como a la facilidad para actualizarse y mantenerse sin distribuir e instalar software a miles de usuarios potenciales.
- Una ventaja significativa es que deberían funcionar igual independientemente del sistema operativo instalado en el cliente.

# Introducción a la Web 2.0

---

## *La web 2.0*

- La web 2.0 significó un cambio radical en el paradigma establecido en la web 1.0
- La web pasa a ser colaborativa → los usuarios dejan su papel de consumidores para convertirse también en productores de contenido.
- Aparecen nuevas tecnologías de desarrollo, como ser las RIA's (Rich Internet Applications).
  - Sin embargo, se sigue usando como base las tecnologías de la web 1.0 y 1.5

# Introducción a la Web 2.0

---

## *RIA (Rich Internet Applications )*

- Son un nuevo tipo de aplicaciones con más ventajas que las tradicionales aplicaciones web, ya que combinan las que ofrecen las aplicaciones Web y las de escritorio.
- La principal ventaja → no se producen recargas de página (trabajo asincrónico), ya que desde el principio se carga toda la aplicación, y sólo se produce comunicación con el servidor cuando se necesitan datos externos como datos de una base de datos o de otros archivos externos.

# Introducción a la Web 2.0

---

## *Características de las RIA*

- Experiencia rica del usuario: se utilizan nuevos conceptos en la web, como los controles para el ingreso de datos.
- Capacidad offline: la aplicación puede funcionar sin necesidad de estar conectada, siempre y cuando no se requiera intercambio de datos con el servidor.
- Productividad alta para el desarrollador: se desarrollaron herramientas que permiten la generación de código de manera automática o más eficiente.

# Introducción a la Web 2.0

---

- Respuesta: las aplicaciones web responden rápidamente y es posible interactuar con la aplicación aún cuando se espera una respuesta del servidor.
- Flexibilidad: es posible cambiar la apariencia, contenidos y servicios de una manera rápida y sencilla.
- Fácil de instalar, distribuir y actualizar: la instalación es igual a la de cualquier aplicación web normal y la actualización de la aplicación consiste solamente en actualizar los archivos en el servidor, incluso cuando hay usuarios conectados.

# AJAX

---

## *Plataformas para desarrollar RIAs → AJAX*

- *AJAX → Asynchronous JavaScript And XML*
- Es una técnica de programación Web que utiliza el objeto XMLHttpRequest → gestiona la entrega de datos entre cliente y servidor a través del protocolo HTTP.
- Permite al navegador solicitar solo la parte de la página web que fue cambiada o actualizada → no hay recarga de la página web.
- Esto reduce el tiempo para actualizar el contenido.
  - Ejemplo: Google Suggests.

# AJAX

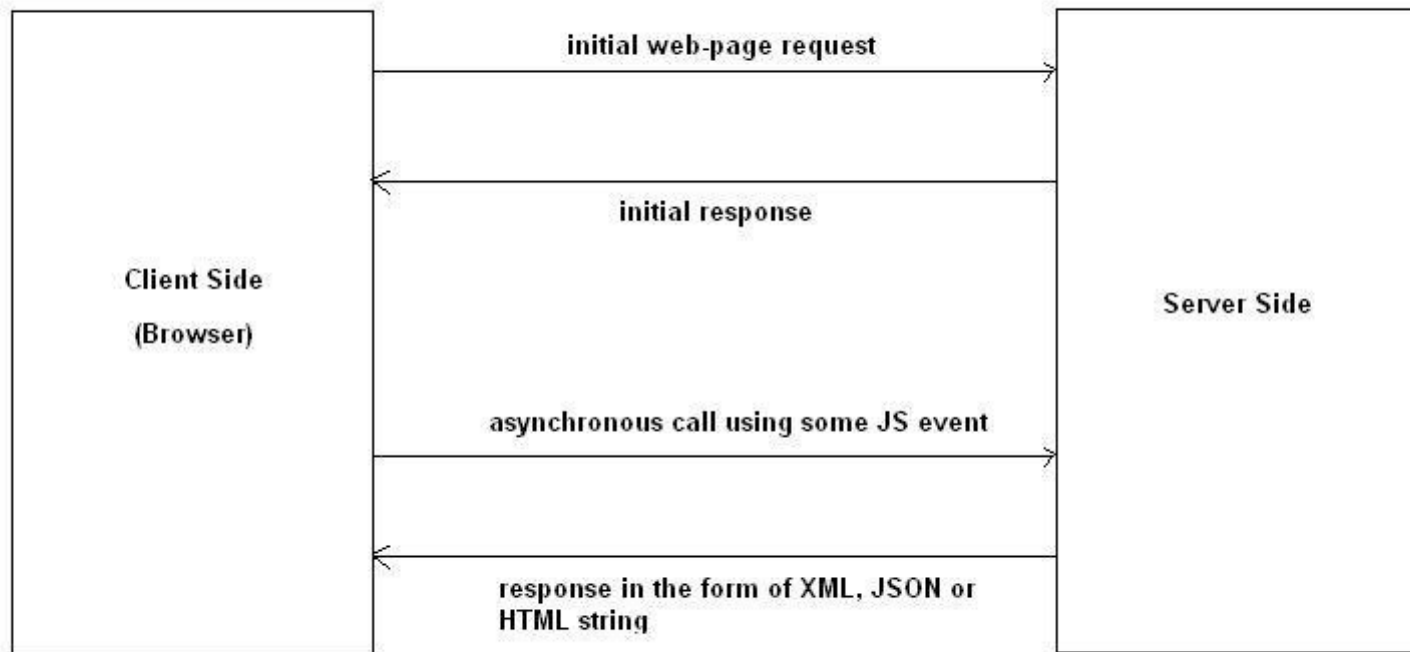
---

- Utiliza:
  - HTML y CSS como lenguajes de estructura y diseño.
  - JavaScript como lenguaje de programación.
  - DOM para trabajar con la estructura del documento web.
  - XML como el formato para el intercambio de datos desde y hacia el servidor.
- Además, requiere que el browser incorpore el objeto XMLHttpRequest para intercambiar los datos.
- No requiere un lenguaje de servidor específico:
  - Lo que importa es que la salida del servidor sea un texto o una salida XML.



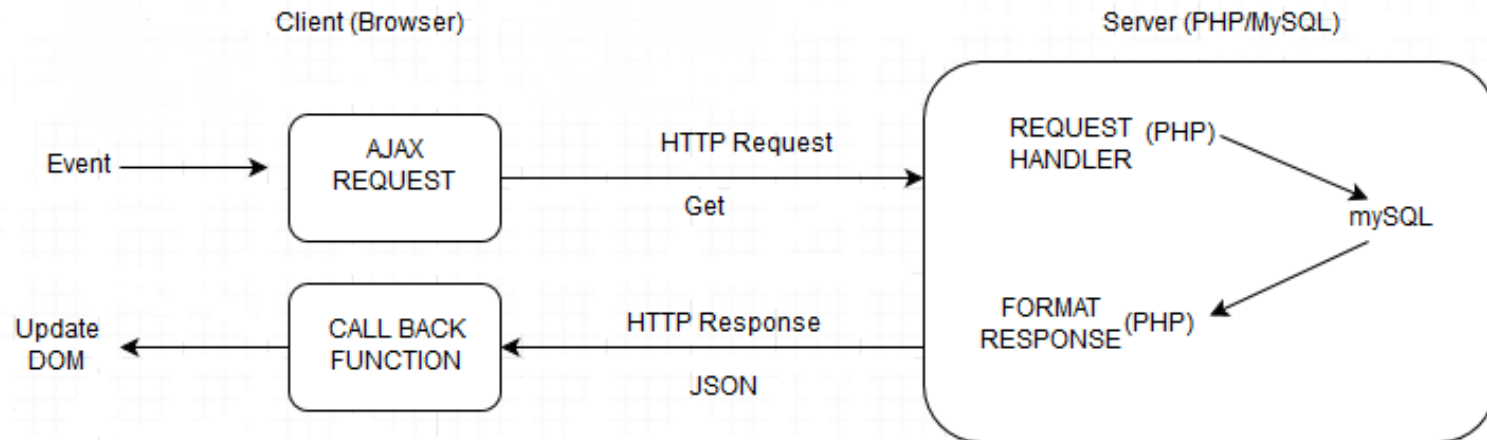
# AJAX

## *Gráficamente*



# AJAX

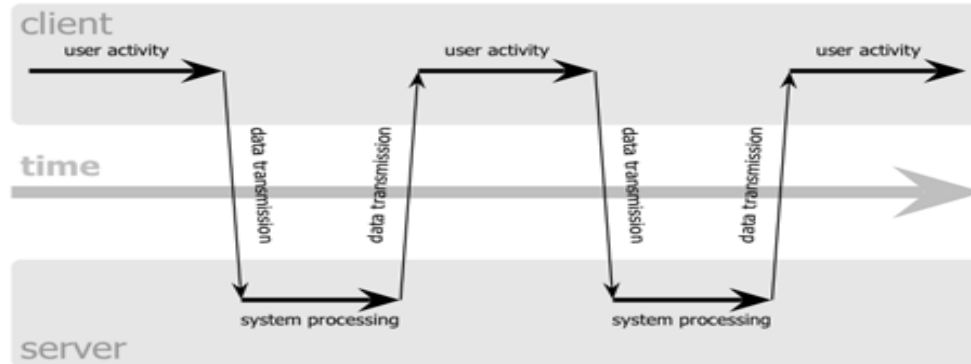
## Gráficamente



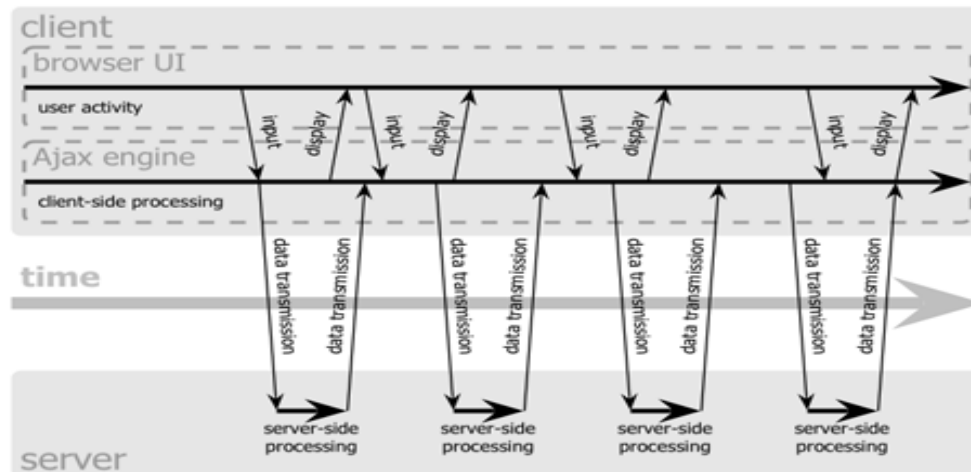
# AJAX

## *Modelo clásico vs. AJAX*

classic web application model (synchronous)



Ajax web application model (asynchronous)



# AJAX

---

## *Casos de uso de AJAX*

- Validación de formularios en tiempo real: nombres de usuario, números de serie, códigos postales, y otros campos se pueden validar contra el servidor antes de que el usuario envíe el formulario completo.
- Auto-Completar: direcciones de correo electrónico, nombres de personas o ciudades se pueden autocompletar mientras el usuario los ingresa.
- Operaciones Maestro – Detalle: basado en eventos del cliente, la página HTML actualiza información detallada de productos según se van seleccionando.

# AJAX

---

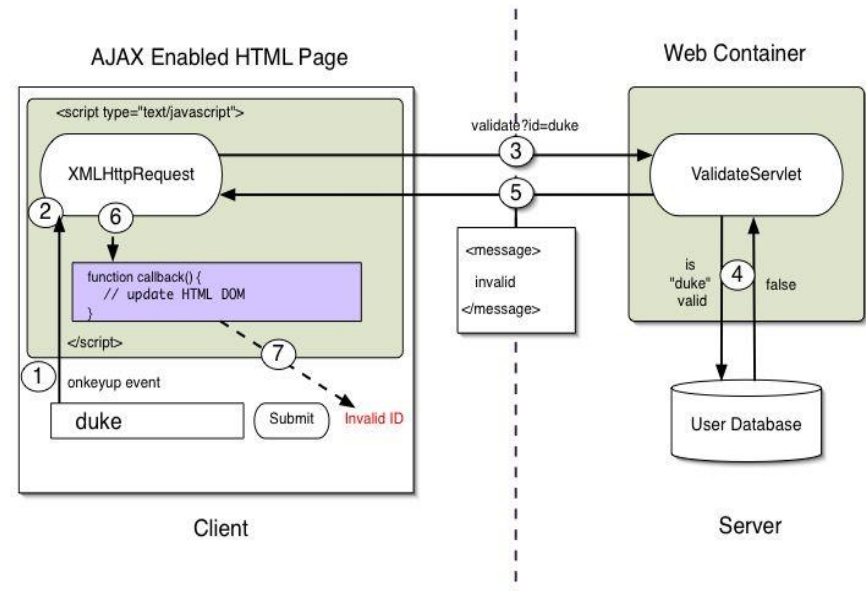
## *El objeto XMLHttpRequest*

- Es la base de las aplicaciones que funcionan con AJAX.
- Es un objeto nativo de JavaScript, con métodos y propiedades específicas.
- Gestiona la comunicación entre el cliente y el servidor, de forma tal que permite generar peticiones con el envío de datos y recepcionar las respuestas.
  - Si es una petición asincrónica, la aplicación web podrá seguir trabajando mientras el servidor procesa y devuelve una respuesta.
  - Si es sincrónica, la aplicación queda a la espera de la respuesta para continuar con su ejecución.

# AJAX

## *Ejemplo de una interacción*

1. Usuario provoca un evento.
2. Se crea y configura un objeto XMLHttpRequest.
3. El objeto XMLHttpRequest realiza una llamada al servidor (petición).
4. La petición se procesa en el servidor.
5. El server retorna un documento XML que contiene el resultado.
6. El objeto XMLHttpRequest llama a la función callback() y procesa el resultado.
7. Se actualiza el DOM de la página asociado con la petición con el resultado devuelto.



# AJAX

---

## *Utilización del objeto XMLHttpRequest*

- Los pasos para utilizar XMLHttpRequest desde un lenguaje de scripts del cliente y poder realizar una petición al servidor son:
  1. Instanciar el objeto XMLHttpRequest
  2. Configurar y abrir la petición
  3. Indicar la función que se encargará de administrar la evolución de la petición (es asincrónica).
  4. Enviar la petición y los datos al servidor
  5. En la función indicada en el punto 3, manipular el estado de la petición, recibir los datos y actuar en consecuencia.

# AJAX

---

- La instanciación en JavaScript se realiza con el operador *new* → Objeto = new XMLHttpRequest()
- La configuración y apertura de la petición se realiza con el método *open* del objeto. Tiene 3 parámetros:
  - Método: método HTTP con el que se hará la petición
  - URL: recurso que se invocará en el servidor
  - Asincronismo: indica tipo petición (*true* para asincrónica y *false* para sincrónica).
- La evolución de una petición se gestiona a través de la propiedad *onreadystatechange*.
  - Es una propiedad que se utiliza para definir el código asociado al evento de cambio de estado en la petición.



# AJAX

---

- Así, a la propiedad *onreadystatechange* debe asignarse una función que se ejecutará automáticamente cada vez que la petición cambie su estado (propiedad *readyState*).
- La propiedad *onreadystatechange* debe definirse antes o después del método *open* y antes del método *send* para que funcione de manera correcta.

# AJAX

---

## *Propiedad readyState*

- Es una propiedad de solo lectura, que contiene un valor entre 0 y 4 que indica el estado de la petición:

Código	Estado	Descripción
0	Sin inicializar	La petición sólo fue instanciada.
1	Cargando	La petición se configuró (con open) pero todavía no se envió.
2	Cargado	La petición se envió pero aún no se recibió respuesta del servidor.
3	Interactivo	El servidor respondió a la petición y están disponibles las cabeceras pero aún falta el contenido
4	Completo	La petición ya finalizó y el contenido está completo

# AJAX

---

- Luego, el envío de la petición se realiza con el método *send*, que tiene un único parámetro opcional (se completa cuando se envían datos con POST).
  - Más allá de ser opcional, se puede incluir con el valor *null* para evitar problemas de compatibilidad en los navegadores.
- Opcionalmente, si se desea abortar una petición, se puede utilizar el método *abort*.

# AJAX

---

## *Propiedad status*

- Es una propiedad de sólo lectura que contiene el código HTTP devuelto por el servidor ante la petición:

Código	Descripción
200	La petición se pudo procesar de manera correcta.
404	La URL o el recurso que se peticiónó no existe en el servidor web.
403	No hay permiso para acceder al recurso en el servidor web.
405	No se acepta el método HTTP. Problema al definir el método GET o POST
500	Error interno del servidor web. El servidor puede estar saturado o hay algún error en el script que se está ejecutando.
503	El servidor no está disponible temporalmente.

# AJAX

---

## *Propiedad responseText*

- Es una propiedad de sólo lectura que devuelve un string con el contenido del cuerpo devuelto por el servidor ante la petición.
  - Los datos son correctos cuando *readystate* = 4 y *status* = 200.
- Se utiliza para recibir del servidor tanto información visual, etiquetas HTML y/o código JavaScript.

# AJAX

## Ejemplo

localhost dice: ✕

Cargado

Aceptar

localhost dice: ✕

Interactivo

☐ Evita que esta página cree cuadros de diálogo adicionales.

Aceptar

localhost dice: ✕

Prueba de AJAX asincronico

☐ Evita que esta página cree cuadros de diálogo adicionales.

Aceptar

```
1 <!-- Codigo_124.htm -->
2 <!DOCTYPE html>
3 <html>
4 <head>
5 <meta charset="iso-8859-1" />
6 <title>AJAX Asincronico</title>
7 <script>
8     function PruebaAJAX()
9     {
10         var peticion = new XMLHttpRequest();
11         peticion.open("GET", "Codigo_114_segundotexto.txt", true);
12         peticion.onreadystatechange = procesarpeticion;
13         peticion.send(null);
14
15         function procesarpeticion()
16         {
17             switch (peticion.readyState)
18             {
19                 case 0: alert("Sin inicializar");
20                     break;
21                 case 1: alert("Cargando");
22                     break;
23                 case 2: alert("Cargado");
24                     break;
25                 case 3: alert("Interactivo");
26                     break;
27                 case 4: if (peticion.status==200)
28                     { //Se proceso la peticion
29                         alert(peticion.responseText);
30                     }
31             }
32         }
33     }
34 </script>
35 </head>
36 <body onload="PruebaAJAX();">
37 </body>
38 </html>
```

# AJAX

## Ejemplo

```
1 <!-- Codigo_116.htm -->
2 <!DOCTYPE html>
3 <html>
4 <head>
5 <meta charset="iso-8859-1" />
6 <title>AJAX Asincronico</title>
7 <script>
8   function ejecutoAJAX()
9   {
10     var peticion = new XMLHttpRequest();
11     peticion.open("GET", "Codigo_116.php?ape=PEREZ&nom=JUANCITO MANUEL EDUARDO", true);
12     peticion.onreadystatechange = procesarpeticion;
13     peticion.send(null);
14
15     function procesarpeticion()
16     {
17       if (peticion.readyState == 4)
18       {
19         if (peticion.status==200)
20         { //Se proceso la peticion
21           alert(peticion.responseText);
22         }
23       }
24     }
25   }
26 </script>
27 </head>
28 <body onLoad="ejecutoAJAX();">
29 </body>
30 </html>
```

```
1 <!-- Codigo_116.php -->
2 <?php
3 echo "<pre>";
4 print_r($_GET);
5 echo "</pre>";
6 ?>
```

localhost dice:

```
<pre>Array
(
    [ape] => PEREZ
    [nom] => JUANCITO MANUEL EDUARDO
)
</pre>
```

Aceptar

# AJAX

---

## JSON

- JSON → JavaScript Object Notation, es una forma muy potente para crear e instanciar objetos en JavaScript, que puede ser utilizada en AJAX.
- Un objeto JSON se encierra entre “{” y “}” y puede contener propiedades y métodos. Los valores de las propiedades pueden ser:
  - Un string
  - Un número
  - Un array
  - Una función (método)
  - Un JSON

```
{  
  Nombre: "Juan",  
  Apellido: "Perez",  
  Edad: 30,  
  Padres: ["Jose", "Maria"],  
  Pareja: { Nombre: "Marta" }  
}
```



# AJAX

---

- JSON puede utilizarse para enviar objetos entre el cliente y el servidor (en ambos sentidos).
- Para el caso del envío desde el servidor hacia el cliente, el objeto JSON se crea y codifica en el script PHP y luego, en el cliente se decodifica con JavaScript.
- Para éstos, se pueden usar funciones y métodos específicos:
  - Para crear un objeto JSON en PHP a partir de un arreglo u objeto: *json\_encode(\$mixed value)*
  - Para decodificar el responsetext recibido en el cliente y convertirlo en JSON con JavaScript: *JSON.parse(peticion.responsetext)*

# AJAX

## Ejemplo: Método GET

```
1 <!-- Codigo_118.php -->
2 <!DOCTYPE html>
3 <html>
4 <head>
5 <meta charset="iso-8859-1" />
6 <title>AJAX Asincronico</title>
7 <script>
8     function buscoPrecio()
9     {
10         var idProducto = document.getElementById("cmbProductos").selectedIndex;
11         var petition = new XMLHttpRequest();
12         petition.open("GET", "Codigo_118a.php?idProducto="+idProducto, true);
13         petition.onreadystatechange = cargoPrecio;
14         petition.send(null);
15
16         function cargoPrecio()
17         {
18             if ((petition.readyState == 4) && (petition.status==200))
19             {
20                 //Se proceso la petition
21                 var myObj = JSON.parse(petition.responseText);
22                 var identificador = document.getElementById("idProducto");
23                 identificador.innerHTML = myObj.idProducto;
24                 var descripcion = document.getElementById("descripcionProducto");
25                 descripcion.innerHTML = myObj.descripcion;
26                 var precio = document.getElementById("precioProducto");
27                 precio.innerHTML = '$ '+myObj.precio;
28             }
29         }
30     }
31 </script>
32 <link rel="stylesheet" href="Codigo_118.css" />
</head>
```

# AJAX

## Ejemplo (continuación)

```
33 <body>
34 <section>
35 <article>
36     Seleccione un producto:
37     <select name="cmbProductos" id="cmbProductos" onChange="buscoPrecio();" >
38         <option value="0">-----</option>
39     <?php
40         include_once("Codigo_118.Producto.class.php");
41         $lstProd = producto::getProductosBD();
42         if (count($lstProd)>0)
43         {   foreach($lstProd as $unProd)
44             {   echo '<option value="'. $unProd->getIdProducto().'">'. $unProd->getDescripcion(). '</option>'; }
45         }
46     <?>
47     </select>
48     <table class="borde">
49     <tr>
50         <td class="borde fondo">Id del Producto</td>
51         <td class="borde"><div id="idProducto">Seleccione un producto</div></td>
52     </tr>
53     <tr>
54         <td class="borde fondo">Descripción</td>
55         <td class="borde"><div id="descripcionProducto">Seleccione un producto</div></td>
56     </tr>
57     <tr>
58         <td class="borde fondo">Precio</td>
59         <td class="borde"><div id="precioProducto">Seleccione un producto</div></td>
60     </tr>
61 </table>
62 </article>
63 </section>
64 </body>
65 </html>
```

# AJAX

## Ejemplo (continuación)

```
1  <?php
2  /*Codigo_118a.php */
3  /* El objeto JSON a enviarse al script JavaScript utilizando AJAX
4  puede generarse a traves de un objeto o a través de un arreglo
5  asociativo de PHP */
6  include_once("Codigo_118.Producto.class.php");
7  $unProdu = producto::getPrecioBD($_GET['idProducto']);
8  if (is_null($unProdu))
9  {   $objTemp = new StdClass();
10     $objTemp->idProducto = 'Producto no encontrado';
11     $objTemp->descripcion = '---';
12     $objTemp->precio = '---';
13     $myJSON = json_encode($objTemp);
14 }
15 else
16 {   $arregloTemp = array('idProducto' => $unProdu->getIdProducto(),
17                         'descripcion' => $unProdu->getDescripcion(),
18                         'precio' => $unProdu->getPrecio());
19     $myJSON = json_encode($arregloTemp);
20 }
21 echo $myJSON;
22 ?>
```

# AJAX

## *Ejemplo (continuación)*

```
1  <?php
2  class producto
3  {
4      private $idProducto;
5      private $descripcion;
6      private $precio;
7
8      public function __construct()
9      {
10
11      }
12
13      public function getIdProducto()
14      { return $this->idProducto; }
15
16      public function setIdProducto($newId)
17      { $this->idProducto = $newId; }
18
19      public function getDescripcion()
20      { return $this->descripcion; }
21
22      public function setDescripcion($newDescripcion)
23      { $this->descripcion = $newDescripcion; }
24
25      public function getPrecio()
26      { return $this->precio; }
27
28      public function setPrecio($newPrecio)
29      { $this->precio = $newPrecio; }
```

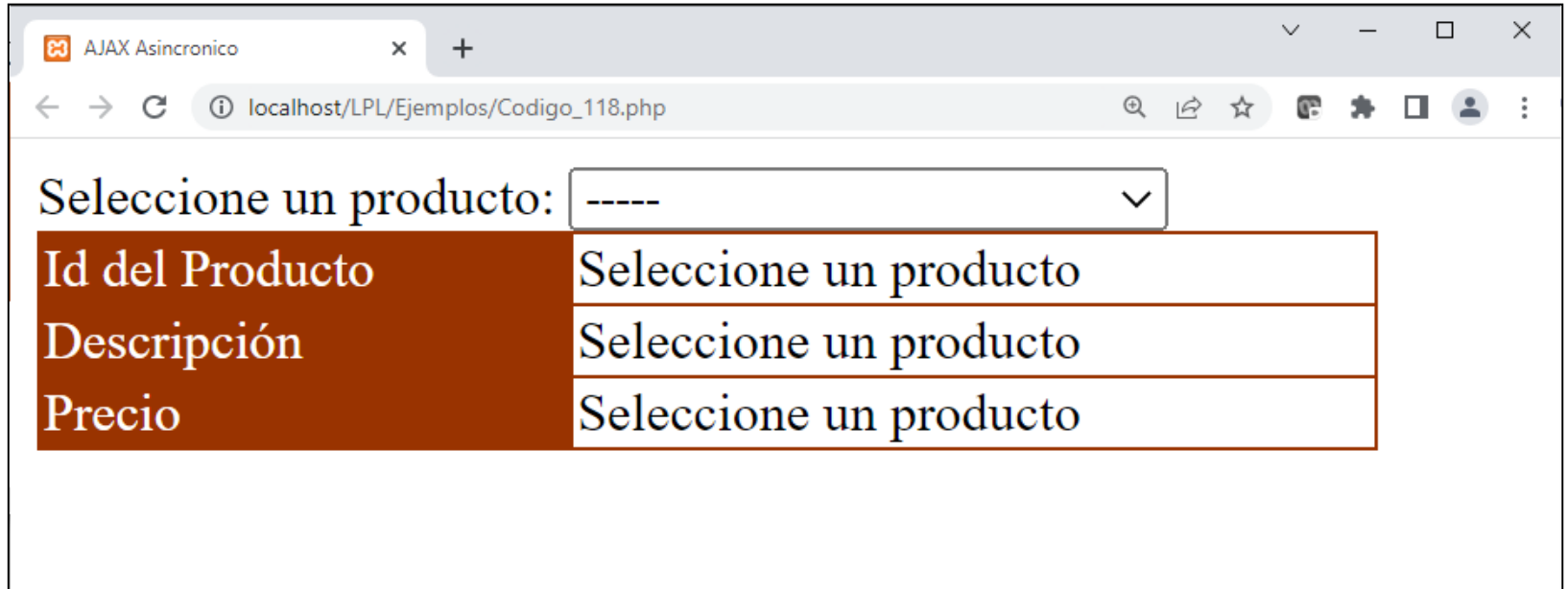
# AJAX

## Ejemplo (continuación)

```
29 public static function getProductosBD()
30 {
31     $listaProductos = array();
32     $con = new mysqli("localhost", "root", "", "chocolates") or die("No es posible conectarse al motor de BD");
33     $consulta = "SELECT * FROM productos ORDER BY idProducto";
34     $listado = $con->query($consulta) or die("No se pudo realizar la consulta");
35     while ($registro = $listado->fetch_object())
36     {
37         $unProducto = new producto();
38         $unProducto->setIdProducto($registro->idProducto);
39         $unProducto->setDescripcion($registro->descripcion);
40         $unProducto->setPrecio($registro->precio);
41         $listaProductos[]=$unProducto;
42     }
43     $listado->free();
44     $con->close();
45     return $listaProductos;
46 }
47
48 public static function getPrecioBD($idProd)
49 {
50     $unProducto = NULL;
51     $con = new mysqli("localhost", "root", "", "chocolates") or die("No es posible conectarse al motor de BD");
52     $consulta = "SELECT * FROM productos WHERE idProducto = ".$idProd." LIMIT 1";
53     $listado = $con->query($consulta) or die("No se pudo realizar la consulta");
54     while ($registro = $listado->fetch_object())
55     {
56         $unProducto = new producto();
57         $unProducto->setIdProducto($registro->idProducto);
58         $unProducto->setDescripcion($registro->descripcion);
59         $unProducto->setPrecio($registro->precio);
60     }
61     $listado->free();
62     $con->close();
63     return $unProducto;
64 }
65 }
66 ?>
```

# AJAX

## *Ejemplo (continuación)*



Seleccione un producto: -----

Id del Producto	Seleccione un producto
Descripción	Seleccione un producto
Precio	Seleccione un producto

# AJAX

## *Ejemplo (continuación)*



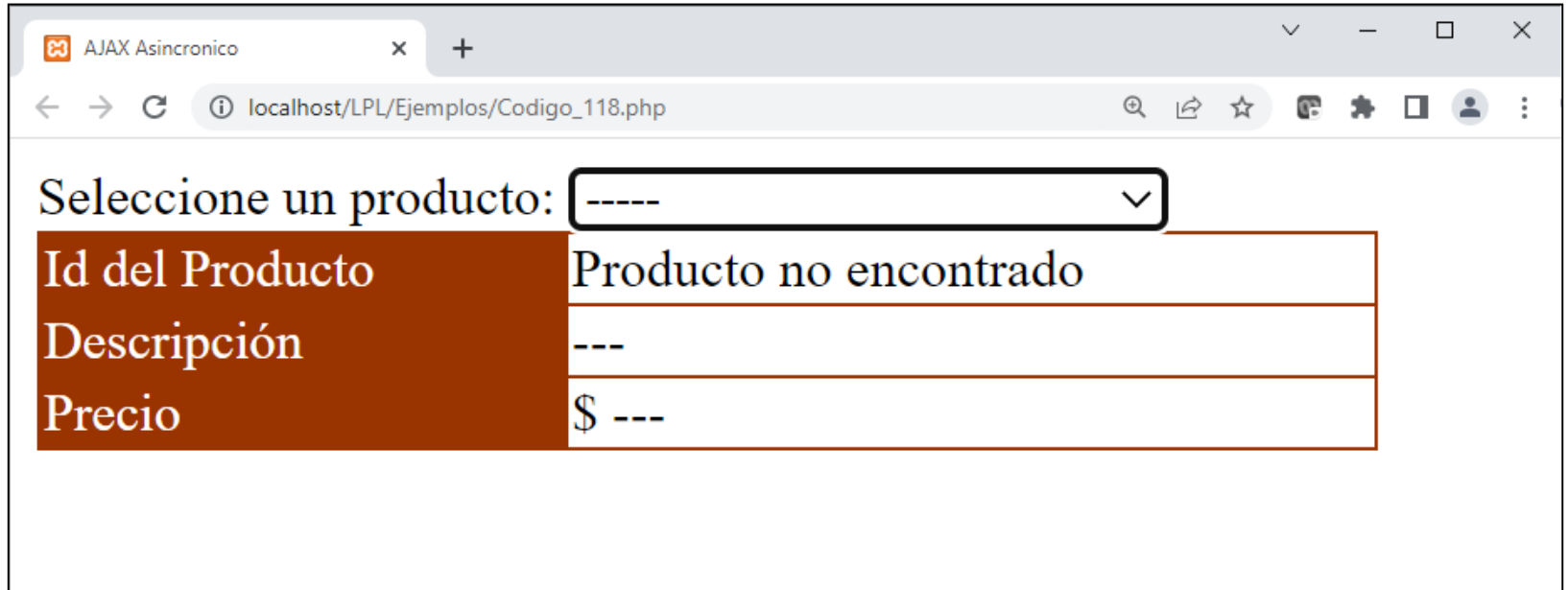
Seleccione un producto: Lindor Milk 100 gramos ▼

Id del Producto	2
Descripción	Lindor Milk 100 gramos
Precio	\$ 1200



# AJAX

## *Ejemplo (continuación)*



Seleccione un producto: -----

Id del Producto	Producto no encontrado
Descripción	---
Precio	\$ ---

# AJAX

## Ejemplo: Método POST

```
1 <!--Codigo_138.php -->
2 <!DOCTYPE html>
3 <html lang="es">
4 <head>
5 <title>AJAX Asincronico</title>
6 <script>
7     function buscoPrecio()
8     {
9         var idProducto = document.getElementById("cmbProductos").selectedIndex;
10        var parametros = "idProducto="+idProducto;
11        var petition = new XMLHttpRequest();
12        petition.open("POST", "Codigo_138a.php", true);
13        petition.onreadystatechange = cargoPrecio;
14        petition.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
15        petition.send(parametros);
16
17        function cargoPrecio()
18        {
19            if ((petition.readyState == 4) && (petition.status==200))
20            { //Se proceso la petition
21                var myObj = JSON.parse(petition.responseText);
22                var identificador = document.getElementById("idProducto");
23                identificador.innerHTML = myObj.idProducto;
24                var descripcion = document.getElementById("descripcionProducto");
25                descripcion.innerHTML = myObj.descripcion;
26                var precio = document.getElementById("precioProducto");
27                precio.innerHTML = '$ '+myObj.precio;
28            }
29        }
30    }
31 </script>
32 <link rel="stylesheet" href="Codigo_118.css" />
33 </head>
```

# AJAX

## Ejemplo (continuación)

```
34 <body>
35 <section>
36 <article>
37     Seleccione un producto:
38     <select name="cmbProductos" id="cmbProductos" onChange="buscoPrecio();">
39         <option value="0">-----</option>
40     <?php
41         include_once("Codigo_118.Producto.class.php");
42         $lstProd = producto::getProductosBD();
43         if (count($lstProd)>0)
44         {   foreach($lstProd as $unProd)
45             {   echo '<option value="'. $unProd->getIdProducto().'">'. $unProd->getDescripcion(). '</option>'; }
46         }
47     ?>
48 </select>
49 <table class="borde">
50     <tr>
51         <td class="borde fondo">Id del Producto</td>
52         <td class="borde"><div id="idProducto">Seleccione un producto</div></td>
53     </tr>
54     <tr>
55         <td class="borde fondo">Descripción</td>
56         <td class="borde"><div id="descripcionProducto">Seleccione un producto</div></td>
57     </tr>
58     <tr>
59         <td class="borde fondo">Precio</td>
60         <td class="borde"><div id="precioProducto">Seleccione un producto</div></td>
61     </tr>
62 </table>
63 </article>
64 </section>
65 </body>
66 </html>
```

# AJAX

## Ejemplo (continuación)

```
1 <?php
2 /* Codigo_138a.php */
3 /* El objeto JSON a enviarse al script JavaScript utilizando AJAX
4 puede generarse a traves de un objeto o a través de un arreglo
5 asociativo de PHP */
6 include_once("Codigo_118.Producto.class.php");
7 if (isset($_POST['idProducto']))
8 {
9     $unProdu = producto::getPrecioBD($_POST['idProducto']);
10    if (is_null($unProdu))
11    {
12        $objTemp = new stdClass();
13        $objTemp->idProducto = 'Producto no encontrado';
14        $objTemp->descripcion = '---';
15        $objTemp->precio = '---';
16        $myJSON = json_encode($objTemp);
17    }
18    else
19    {
20        $arregloTemp = array('idProducto' => $unProdu->getIdProducto(),
21                             'descripcion' => $unProdu->getDescripcion(),
22                             'precio' => $unProdu->getPrecio());
23        $myJSON = json_encode($arregloTemp);
24    }
25    else
26    {
27        $arregloTemp = array('idProducto' => 'Producto inexistente',
28                             'descripcion' => '---',
29                             'precio' => '---');
30        $myJSON = json_encode($arregloTemp);
31    }
32    echo $myJSON;
33 ?>
```

# AJAX

---

## *Listado de comprobaciones – Puntos de control*

Ante ocurrencia de errores se debe chequear:

1. Que la petición se haya armado correctamente → imprimir la cadena de parámetros
2. Que la petición llego correctamente al servidor web → imprimir en la consola
3. Que el servidor web obtiene correctamente la información solicitada → imprimir en la consola
4. Que el servidor envía la información en el formato JSON correctamente → imprimir en la consola
5. Que el cliente recibe correctamente la respuesta en formato JSON → imprimir el responseText

**Fin del curso 2024**

**Muchas gracias!!**