Investigación

AJAX

versión 1.0

Preparado por:

José Evelio Castro Quesada 2016086968

José Gómez Casasola 2016095929

Angelo Ortega Ramírez 2017080055

Alejandro Tapia Barboza 2016167784

6 de Enero del 2020

Introducción

Cuando se realiza una página web, son muchas las herramientas que tenemos a nuestro alcance para realizar diferentes funciones. Cada una de ellas se especializa en diferentes funciones y dependiendo de lo requerido se puede utilizar una o otra. Una de ellas, es AJAX y en este documento se realizará una pequeña investigación en torno a esta herramienta. Se expondrá un poco de su historia, su fin de creación y el contexto de esta, así como sus principales funcionalidades, además se definirán algunas de sus ventajas así como desventajas a la hora de usarlo. También se explicará cuales son los navegadores que la soportan y por último se explicará un pequeño taller, para que realizandolo se entienda mejor la herramienta.

AJAX

AJAX significa Asynchronous JavaScript And XML, la cual es una técnica de desarrollo web para la elaboración de aplicaciones interactivas o RIA (Rich Internet Applications). Este tipo de aplicaciones se ejecutan en el cliente mientras mantiene una comunicación asíncrona, en segundo plano, con el servidor. Es una técnica válida para múltiples plataformas, sistemas operativos y navegadores ya que está basado en estándares abiertos como JavaScript y Document Object Model (DOM).

Con esta técnica es posible realizar cambios sobre las páginas sin necesidad de recargarlas, con lo que se incrementa la velocidad, usabilidad e interactividad en las aplicaciones web. Es una tecnología asíncrona, en el sentido de que los datos adicionales se solicitan al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página.

Las funciones de llamada de AJAX se ejecutan comúnmente mediante JavaScript, mientras que el acceso a los datos mediante XMLHttpRequest.

Historia

AJAX surge de una necesidad de obtener datos de manera dinámica en páginas web. Anteriormente se tenía que recargar la página cada vez que se realizaba una consulta a base de datos, proceso que era sumamente ineficiente dado que incluso los datos estáticos de la página web eran recargados. AJAX se aprovecha de la introducción de tags

con capacidades asíncronas de HTML como el iframe o el object. El objeto XMLHttpRequest fue creado por el equipo de Microsoft Outlook Web Access, este apareció por primera vez en Internet Explorer 5.0 en 1999. Luego fue implementado por Mozilla, Safari y Opera. Inicialmente no se le sacaba mucho provecho a las capacidades HTTP asíncronas. No fue hasta 2004 que Google desplegó una versión compatible con distintos navegadores basándose en estándares. Se demostró su potencial con Gmail y con Google Maps. El término AJAX fue acuñado en 2005 por Jesse James Garret. Un año después, el World Wide Web Consortium publicó la primera versión de una estandarización del objeto XMLHttpRequest.[3]

Tecnologías incluidas

AJAX utiliza la combinación de las siguientes tecnologías:

- HTML y CSS para el diseño que presenta la información (interfaz gráfica)
- Document Object Model, accedido con lenguajes como JavaScript y JScript, con el fin de poder interactuar y mostrar dinámicamente la información presentada.
- El objeto XMLHttpRequest para realizar el intercambio asíncrono de datos con el servidor web.
- XML, es generalmente utilizado para transferir los datos al servidor, aunque también se puede utilizar texto plano, JSON y hasta EBML.

Ventajas

- Mejor experiencia de usuario. Ajax permite que las páginas se modifiquen sin tener que volver a cargarse, dándole al usuario la sensación de que los cambios se producen instantáneamente. Este comportamiento es propio de los programas de escritorio a los que la mayoría de los usuarios están más acostumbrados. La experiencia se vuelve mucho más interactiva.
- Optimización de recursos. Al no re-cargarse la página se reduce el tiempo implicado en cada transacción. También se utiliza menos ancho de banda.
- Alta compatibilidad. Ajax es soportado por casi todas las plataformas Web.

Desventajas

- Problemas de acceso. Normalmente, si un usuario refina una consulta a una base de datos a través de muchos criterios (por ejemplo, categoría, precio, forma de pago, etc.), la página se recargará con una URL que reflejará los parámetros ingresados. El usuario puede guardar esa URL para volver a acceder a los resultados ya filtrados fácilmente. Pero con Ajax la URL no se modifica ante la consulta, por lo que deberemos volver a ingresar cada filtro manualmente cuando queramos recuperar los resultados deseados. Existen métodos para modificar este comportamiento, pero agregan dificultad al desarrollo y peso al sitio.
- Problemas de SEO. Los buscadores tienen dificultades al analizar el código escrito en JavaScript. El hecho de que se no se generen nuevas URL elimina un importante factor de posicionamiento.
- Dificultad. Las aplicaciones con Ajax suelen requerir de un mayor tiempo de desarrollo.

Soporte de AJAX

Navegadores que lo soportan

- Mozilla
- Safari
- Chrome
- Opera (8 en adelante)
- Internet Explorer (5 en adelante)
- Edge

Navegadores que no lo soportan

- Opera (anteriores a 7)
- Explorer (anteriores a 5)
- Safari (anteriores a 1.2)
- Dillo

Sintaxis AJAX

Para **JavaScript** se hace uso del objeto «XMLHttpRequest» es un objeto de JavaScript ampliamente utilizado en programación AJAX que puede ser usado para recibir cualquier tipo de dato (no solamente XML; como su nombre sugiere), y admite otros formatos además de HTTP, incluyendo file y ftp.

Se instancia de la siguiente manera:

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

Entre sus métodos encontramos:

- open(method, url): Inicializa una petición. Su parámetro método puede ser de tipo "GET", "POST", "PUT", o "DELETE". Su parámetro url es un string que representa el url donde enviar la petición.
- send(body): Envía la solicitud al servidor. Si la solicitud es asíncrona (que es la predeterminada), este método vuelve tan pronto como se envía la solicitud, si la solicitud es sincrónica, este método no regresa hasta que llegue la respuesta. El parámetro *body* corresponde a un cuerpo de datos para enviar en la solicitud XHR.

Algunos atributos de utilidad son:

- responseText: un atributo de tipo AString, que corresponde a la respuesta al pedido, como texto o *null* si el pedido no fue exitoso o todavía no se envió.
- readyState: un atributo de tipo ***long*****,** que retorna un número entre 0 a 4, según el estado del pedido de acuerdo a la *Tabla 1*.

Valor	**Estado**	**Descripción**
0	UNINITIALIZED	todavía no se llamó a open()
1	LOADING	todavía no se llamó a send()
2	LOADED	send() ya fue invocado, y los encabezados y el estado están disponibles
3	INTERACTIVE	Descargando; responseText contiene información parcial.

4	COMPLETED	La operación está terminada.

Tabla 1

- onreadystatechange: un atributo de tipo ***nsIDOMEventListener***. Corresponde a una función del objeto JavaScript que se llama cuando el atributo readyState cambia.

Ejemplo de AJAX

Ejemplo Hello World

Se hará un ejemplo «*Hello World*» en la que los archivos en cuestión pertenecen obligatoriamente al mismo servidor, es decir, no se podrá hacer uso de información de terceros en nuestro sitio. Además, se podría presentar problemas parciales o totales en navegadores antiguos y navegadores de la familia de Internet Explorer.

Los pasos previos antes de codificar el ejemplo son:

1) Descargar e instalar Node.js para poder crear un servidor local (Puede ser otro entorno)
2) Instalar el paquete http-server ─se recomienda de manera global─ con el comando: *npm install -g http-server*.
3) Crear un archivo con extensión .txt (funcional para este ejemplo) y escribir en el: “Hola mundo”.
4) Crear un archivo con extensión .html.
5) Contener ambos archivos en la misma carpeta (ver imagen 1).

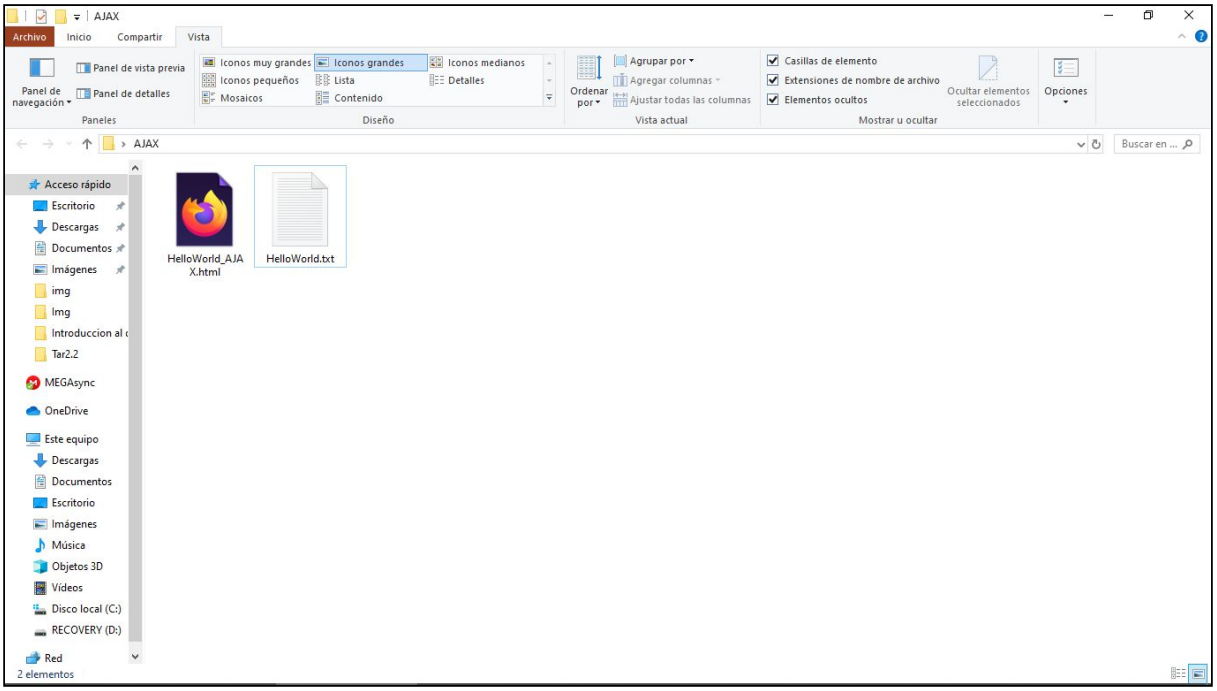


Imagen 1

Indicaciones para el .html:

1) Indicar los tag: <html></html>
2) Dentro de los tag <html></html>, dar formato al título agregando: *<html><head><title>*Ejemplo-AJAX</head>
3) Dentro de los tag <html></html>, agregar: <body><h2>Ejemplo con AJAX</h2><script></script> (Ver imagen 2)
4) Dentro de los tag <script></script> (práctica no recomendada dentro de un archivo html) se realizan las siguientes acciones:
    a) Instanciar un objeto XMLHttpRequest:
    connection = new XMLHttpRequest();
    b) Indicar un manejador de eventos para la información entrante:
    connection.onreadystatechange = function() {
        if (this.readyState == 4) { //El 4 indica operación finalizada.
            alert(this.responseText);
        }
    }
    c) Hacer la llamada:
    connection.open("GET", "HelloWorld.txt");
    connection.send(null);

(Ver imagen 3)

5) En la carpeta donde se contienen los archivos se abre una terminal y se corre un servidor con el comando: *http-server*.
6) En el navegador de preferencia se busca: *http://localhost:8080/*, y se abrirá la página tal como se muestra en la imagen 4, concluyendo el ejemplo.

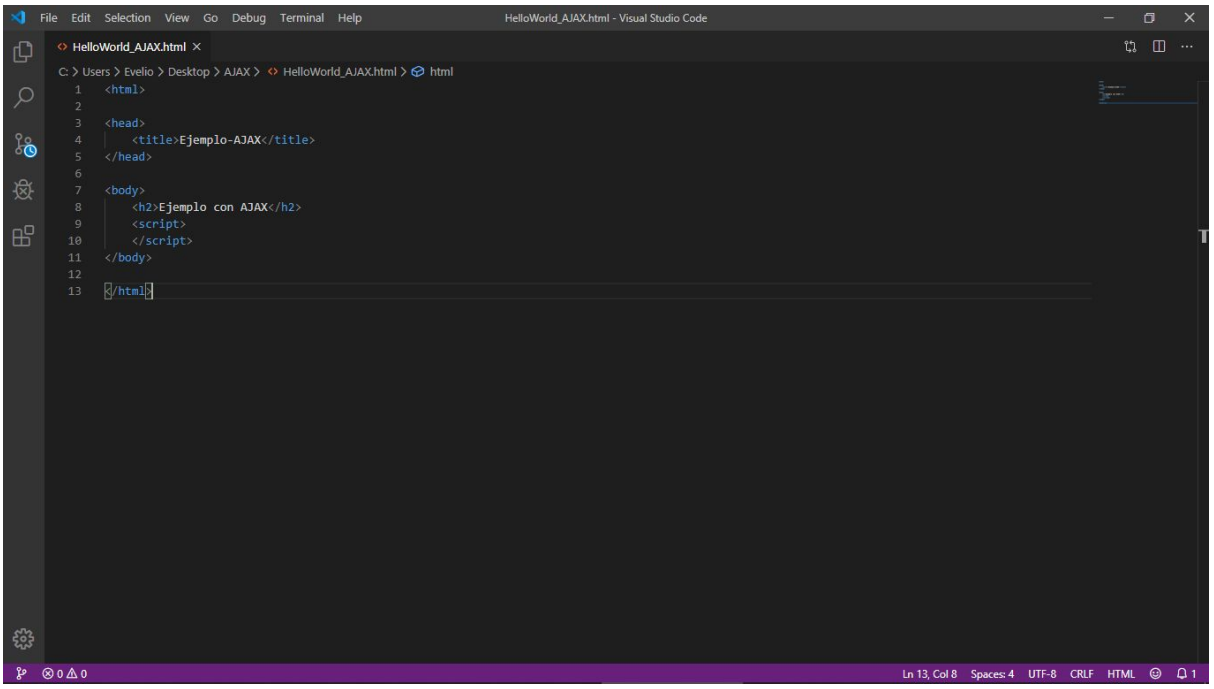


Imagen 2

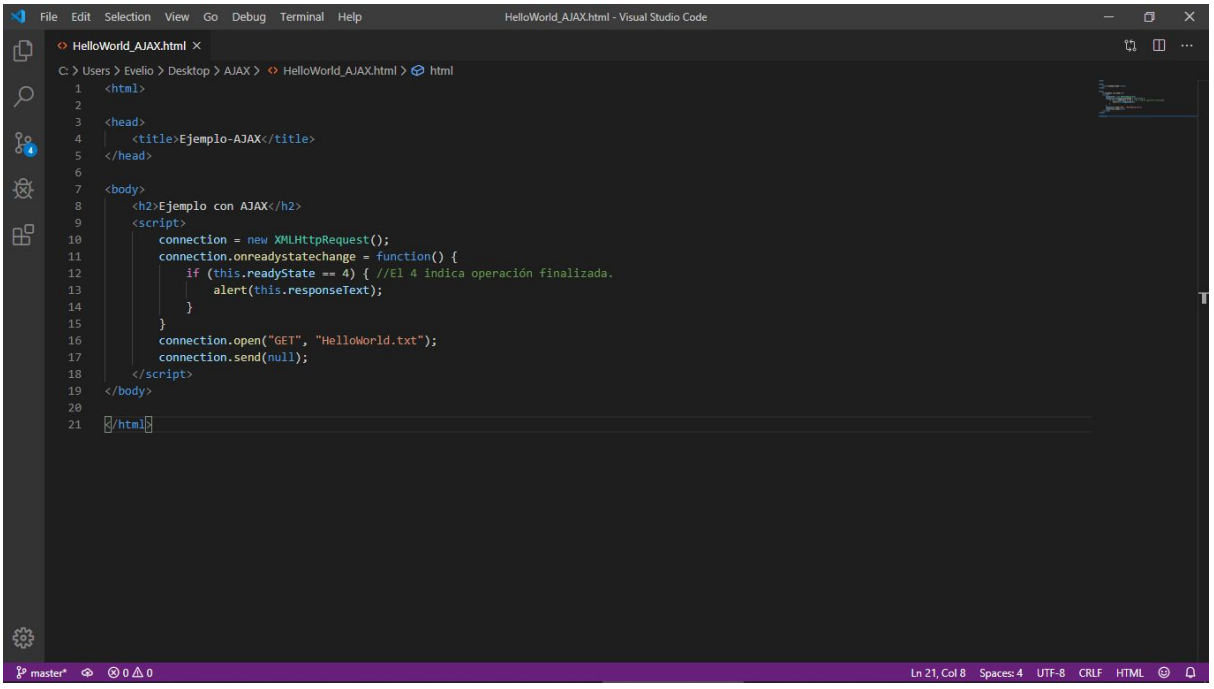


Imagen 3

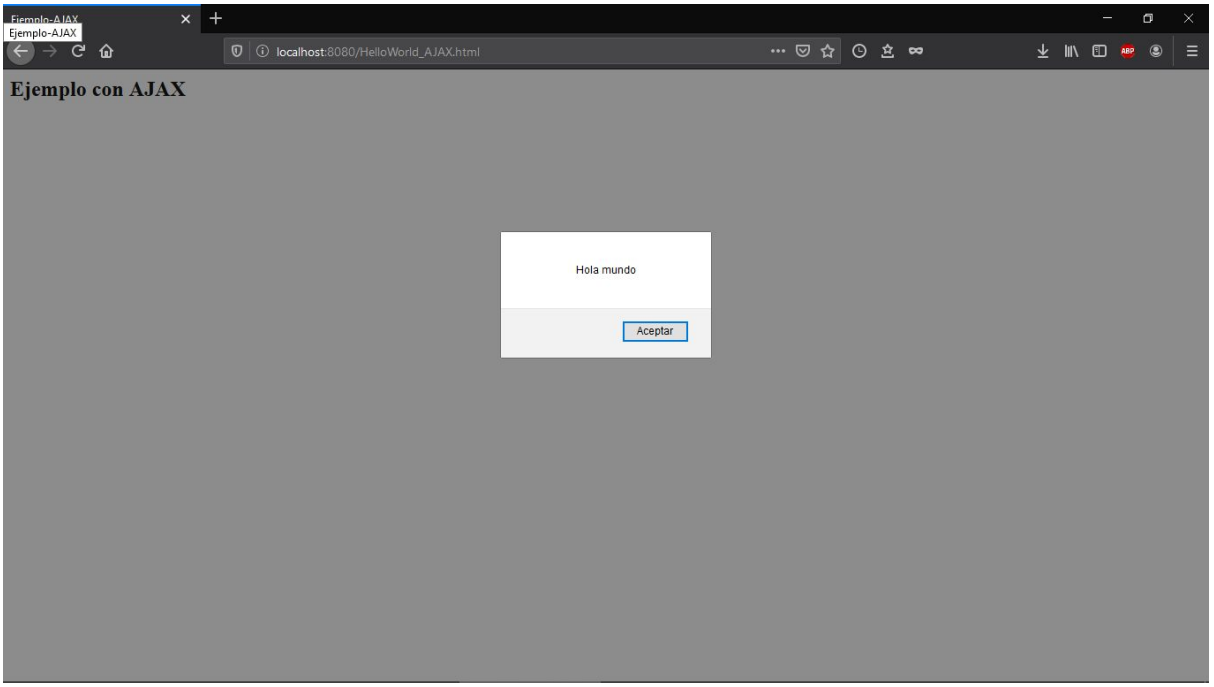


Imagen 4

Taller

Ejercicio de AJAX utilizando JQuery

Se hará un ejemplo utilizando el método load de Jquery: `$(selector).load(url[, data][, complete])` donde *selector* es el elemento al que le queremos añadir los datos, *url* es el url de dónde provienen los datos. *Data* es el texto plano que se enviará al servidor de origen con el request y *complete* es la función de callback que se llamará una vez se complete la solicitud. Todos los archivos en cuestión pertenecen obligatoriamente al mismo servidor, es decir, no se podrá hacer uso de información de terceros en nuestro sitio. Además, se podría presentar problemas parciales o totales en navegadores antiguos y navegadores de la familia de Internet Explorer.

Los pasos previos antes de codificar el ejemplo son:

1) Descargar e instalar Node.js para poder crear un servidor local (Puede ser otro entorno)
2) Instalar el paquete http-server ─se recomienda de manera global─ con el comando: *npm install -g http-server*.
3) Crear un archivo index.html.
4) Crear un archivo index.js.
5) Crear un archivo bio.txt

6) Contener todos los archivos en la misma carpeta (ver imagen 1).

Imagen 1

Indicaciones para el .html:

1) Indicar los tag: <!doctype html>
<html></html>

2) Dentro de los tag <html></html>, dar formato al título agregando, incluir las hojas de estilo y el script de jquery:

```
<head>
    <meta charset="utf-8">

    <title>AJAX Example</title>
    <!-- Custom styles for this template -->
    <link href="./index.css" rel="stylesheet">
    <script type="text/javascript" src=
"https://cdnjs.cloudflare.com/ajax/libs/jquery/3.1.1/jquery.min.j
s" ></script>
</head>
```

3) Dentro de los tag <html></html>, agregar:

```
<body>
<h3>An AJAX Example <span>with jQuery's <code>load</code></span>
method</h3>
<div>
  <img
src="https://s3-us-west-2.amazonaws.com/s.cdpn.io/162656/Einstein
.jpg" alt="Einstein">
  <button id="request">Learn more about Einstein</button>
  <div id="bio"></div>
</div>
<script type="text/javascript" src="./index.js"></script>
</body>
```

Esto le dará forma a nuestro archivo y cargará el archivo .js al que le añadiremos la lógica para realizar la carga del archivo bio.txt.

4) Ahora pasamos al archivo index.js para añadir la lógica.
5) Creamos dos variables para referenciar el botón que hace la petición y dónde en el html pondremos el resultado:

```
var $btn = $('#request');
var $bio = $('#bio');
```

6) Añadimos el listener al *click* del botón

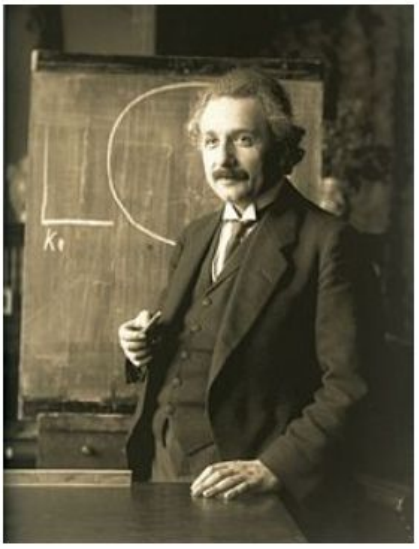
```
$btn.on('click', function() {
  $(this).hide();
  $bio.load('bio.txt', completeFunction);
});
```

7) creamos la función de callback para verificar resultados:

```
function completeFunction(responseText, textStatus, request) {
  $bio.css('border', '1px solid #e8e8e8');
  if(textStatus == 'error') {
       $bio.text('An error occurred during your request: ' +  request.status +
' ' + request.statusText);
  }
}
```

8) Ahora cuando un error ocurra, esto será desplegado al usuario.
9) En la carpeta donde se contienen los archivos se abre una terminal y se corre un servidor con el comando: *http-server*.
10) En el navegador de preferencia se busca: *http://localhost:8080/*, y se abrirá la página tal como se muestra en las imágenes 3 y 4, concluyendo el ejemplo.

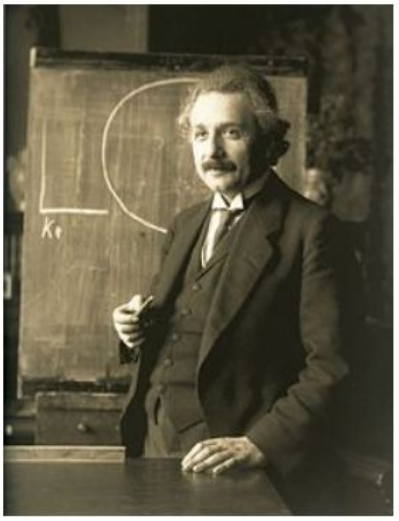
An AJAX Example With JQuery's Load Method



Learn more about Einstein

Imagen 3

An AJAX Example With JQuery's `Load` Method



Albert Einstein (14 March 1879-18 April 1955) was a German-born theoretical physicist. He developed the general theory of relativity, one of the two pillars of modern physics (alongside quantum mechanics). Einstein's work is also known for its influence on the philosophy of science.

Einstein is best known in popular culture for his mass-energy equivalence formula E = mc2 (which has been dubbed "the world's most famous equation"). He received the 1921 Nobel Prize in Physics for his "services to theoretical physics", in particular his discovery of the law of the photoelectric effect, a pivotal step in the evolution of quantum theory.

Imagen 4

Bibliografía

[1] Anyelguti, A. (2018, 13 julio). Qué es Ajax. Recuperado 18 diciembre, 2019, de https://aprende-web.net/progra/ajax/ajax_1.php

[2] Emiley, J. (2016, 27 noviembre). ¿Qué es AJAX? Recuperado 18 diciembre, 2019, de https://www.digitallearning.es/blog/que-es-ajax/

[3] Digital Learning. (2017, 14 abril). The History of AJAX. Recuperado 18 diciembre, 2019, de https://java-samples.com/showtutorial.php?tutorialid=238

[4] Martsoukos, G. (2016, 9 febrero). A Beginner's Guide to AJAX With jQuery. Recuperado 18 diciembre, 2019, de https://webdesign.tutsplus.com/tutorials/a-beginners-guide-to-ajax-with-jquery--cms-25126

[5] Universidad de Alicante. (s.f.). AJAX. Servicio de Informática Mootools. Recuperado 18 diciembre, 2019, de https://si.ua.es/es/documentacion/mootools/ajax.html