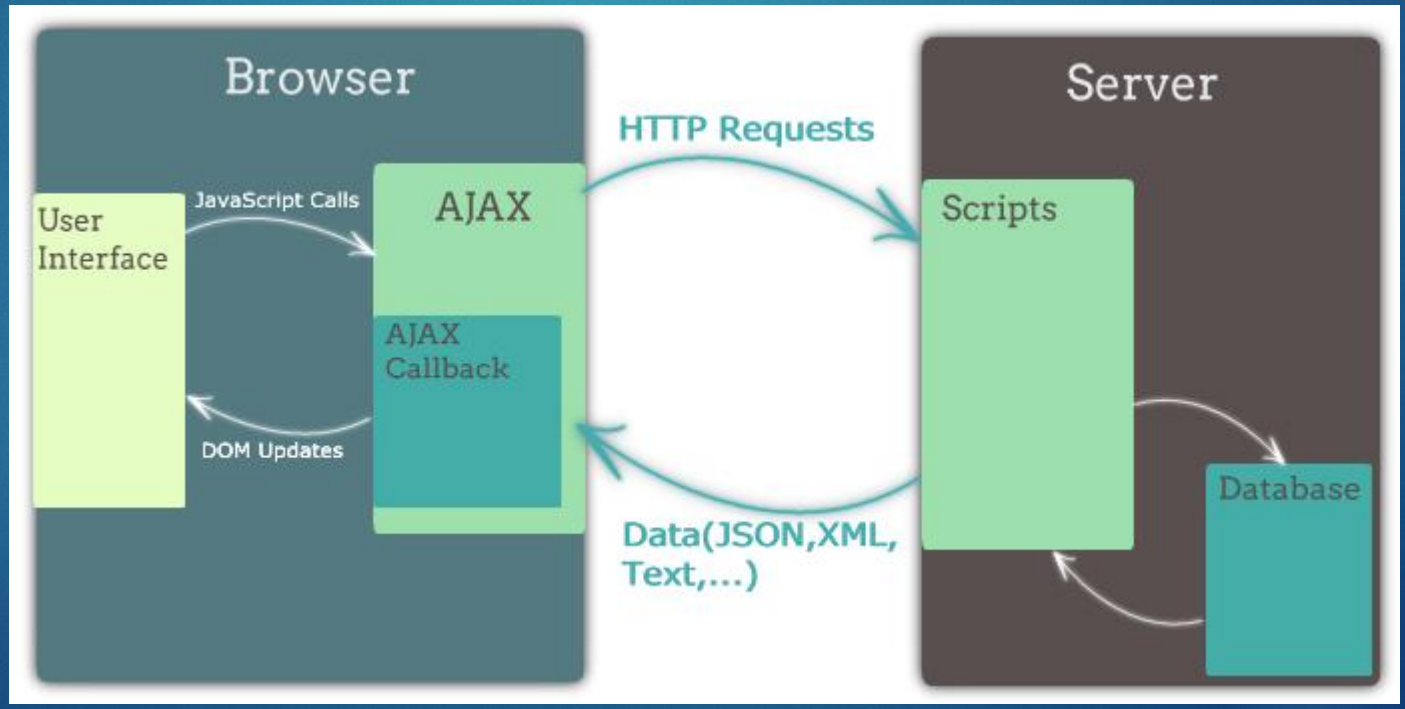
Ajax

LABORATORIO DE COMPUTACIÓN III

UTN-FRA

Qué es Ajax?

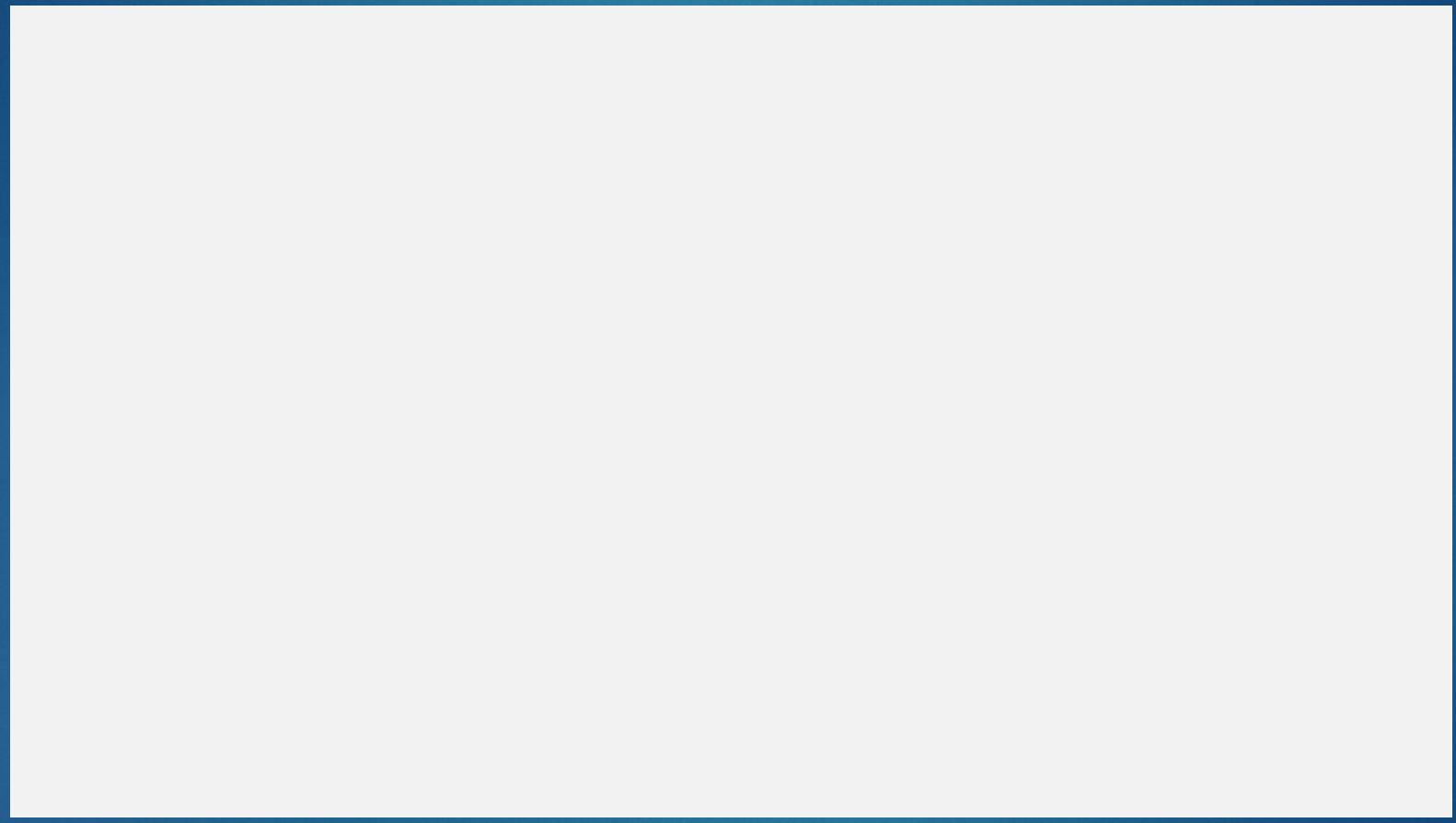
- AJAX es un acrónimo que significa JavaScript asíncrono y XML. Es una tecnología de programación que se utiliza para crear páginas web más interactivas. Usando AJAX, puede crear páginas web que pueden actualizar su contenido sin recargar. AJAX permite que una página web se comunique directamente con el servidor, recupere información y se actualice. Todo esto sucede sin que la página se vuelva a cargar.



Entendiendo Ajax

- Para entender AJAX, hay dos cosas que debes saber desde el principio. En primer lugar, AJAX no es un lenguaje de programación. Tampoco es un software. AJAX es un paradigma de programación, una técnica. Es una técnica para usar tecnologías web que incluyen CSS, HTML, JavaScript, DOM y XML o JSON. Básicamente, para usar AJAX en sus aplicaciones web, necesita tener una comprensión básica de esas tecnologías.
- En segundo lugar, necesita una comprensión básica de cómo los navegadores web interactúan con un servidor web. Tradicionalmente, los navegadores web interactúan con un servidor web de manera síncrona. La interacción se realiza principalmente en tres pasos, repetidos una y otra vez: 1. Un navegador web solicita una página del servidor 2. El servidor envía la página al navegador 3. El navegador muestra la página al usuario
- Para este modelo tradicional, el trabajo de un desarrollador web es simple. Diseñe sus páginas y enlácelas con la etiqueta de anclaje (<a>). Tan pronto como un usuario haga clic en cualquier enlace, es el navegador el que enviará una solicitud al servidor y cargará la página correspondiente. El desarrollador web no solicita directamente información del servidor.

Modelo tradicional Vs Modelo Ajax



Modelo Ajax

- En el modelo AJAX, según las acciones del usuario (haciendo clic en un botón o imagen, por ejemplo), el código JavaScript envía la solicitud al servidor web, recibe la respuesta y actualiza la página web.

- Un buen ejemplo de AJAX en acción son las páginas de Facebook o Twitter. Cuando te desplazas hacia abajo en tu página de Facebook, seguís obteniendo tus feeds de noticias anteriores y mostrándolos. Hace esto sin volver a cargar la página. Tienes la impresión de desplazarte hacia una página web interminable y muy larga.

- Un modelo de solicitud AJAX no es tan fácil de implementar como el modelo tradicional. Requiere que un desarrollador web realice alguna programación de JavaScript en la página web. El JavaScript envía una solicitud al servidor; lee la respuesta y muestra los resultados actualizando el DOM de la página. Esto puede sonar complicado, pero en realidad no lo es. Los navegadores modernos tienen un objeto incorporado llamado objeto XMLHttpRequest. Este objeto hace que sea bastante fácil para JavaScript comunicarse con el servidor.

Ajax con Javascript puro

- Podemos usar el objeto XMLHttpRequest (también conocido como XHR) para comunicarnos con el servidor. Usando el objeto XHR, una página web puede interactuar con scripts PHP, NODEJS, bases de datos, aplicaciones e incluso archivos de texto ubicados en el servidor. Por "interactuar" queremos decir que puede enviar y recuperar datos de esas diversas fuentes.

Esta interacción generalmente es impulsada por JavaScript, y una implementación simple de Ajax tiene lugar en cinco pasos:

- Crear una instancia del objeto XMLHttpRequest
- Use el método open () de XHR para especificar qué tipo de datos desea
- Crear una función para utilizar los resultados.
- Use el método send () de XHR para enviar la solicitud
- Recibe la respuesta

Primera aplicación con Ajax

- Nuestra aplicación hará algo muy básico. Leerá el contenido de un archivo de texto del servidor y lo usará para reemplazar el texto actualmente en la página. Llevará a cabo esta acción con solo hacer clic en un botón.
- Entonces, tendremos dos cosas: nuestra página web simple y un archivo de texto llamado “documento.txt". Este archivo de texto estará ubicado en la misma carpeta que nuestra página web.

```
<div id="info">
    <h2> Texto que se reemplazará con Ajax </h2>
</div>
<button id='btn'> Cambiar Texto </button>
```

Index.html

```
<h3> Hola, Bienvenido al mundo de Ajax</h3>
<p> Ajax es <b>super fácil </b>, No?
```

Documento.txt

Codeando la aplicación

- Nuestro objetivo de programación es simple: al cargar la página, el usuario debe ver el contenido en “info". Al hacer clic en el botón, el contenido debe cambiar y mostrar lo que está en el archivo de texto. Ahora, comencemos.
- Primero, necesitamos un nombre para nuestra función, llamémosla “cargarNuevoTexto ()". Esta función llevará a cabo cada paso en nuestro objetivo de programación, es decir, leer el contenido en “documento.txt" y usarlo para reemplazar el texto en “info".
- Ahora, podemos agregar un evento onclick al botón. Nuestro botón se convierte en:

```
<button id='btn' onclick="cargarNuevoTexto()"> Cambiar Texto </button>
```

- Básicamente, hacer clic en el botón debe ejecutar nuestra función, la función "cargarNuevoTexto()" y, por lo tanto, cambiar el texto de la página.

Todo listo. Ahora Ajax

- 1. Cree una instancia del objeto XMLHttpRequest
- Para crear una instancia de XHR, simplemente obtiene un nombre de variable y utiliza el nuevo método XMLHttpRequest () para crear la instancia. Llamemos a nuestra instancia "xhr". Entonces, crearemos una instancia de la siguiente manera :

```
let xhr = new XMLHttpRequest();
```

- 2. Utilice el método open () de XHR para especificar qué tipo de datos desea
- Open () de XHR se utiliza para especificar el tipo de datos que un objeto desea del servidor. Básicamente lo usa para describir lo que desea del servidor. Toma tres argumentos, es decir, el tipo de solicitud, la ubicación del archivo en el servidor y un indicador síncrono. Llamarlo se ve así:

xhr.open(request, url, async);

- Se desglosan de la siguiente manera:

- **solicitud** : este es el tipo de solicitud que está enviando al servidor. Toma uno de dos valores: GET o POST. En términos simples, GET es para recuperar algo del servidor. POST es para enviar algo al servidor.
- **url** : esta es la URL del archivo en el servidor. Puede ser una URL estática o relativa o simplemente la ruta desde la carpeta que contiene la página web.
- **asíncrono** : se utiliza para determinar si la solicitud debe ejecutarse de forma asíncrona o no. Toma el valor "verdadero" o "falso". Verdadero es para ejecución asincrónica. Falso es para ejecución síncrona.
- En nuestro caso, llamaremos al método open () de la siguiente manera:

```
xhr.open("GET", "documento.txt", true);
```

- 3. Cree una función para recibir y utilizar los resultados.
- Un objeto XHR tiene muchas variables incorporadas en las que almacena los datos recuperados del servidor. Una de estas variables se llama responseText. Ahora, responseText generalmente contiene cualquier información de texto recuperada del servidor.
- Cuando llamamos a xhr.open(), buscará la información de texto almacenada en “documento.txt" y la almacenará en su variable responseText. Entonces, para acceder a los datos, simplemente tenemos que llamar a xhr.responseText
- Dado que nuestro objetivo es reemplazar el <h2> en con el nuevo texto leído desde el servidor, usamos document.getElementById ().

Entonces, crearemos una función anónima de la siguiente manera:

```
function()
{
    document.getElementByID("info").innerHTML = xhr.responseText;
}
```

Esta función básic texto obtenido del servidor. ¿Dónde llamaremos a esta función? Llegaremos a eso en breve.

- 4. Use el método send () de XHR para enviar la solicitud

El método send () se usa para enviar la solicitud al servidor. No toma ningún parámetro, por lo que simplemente lo llama de la siguiente manera:

```
xhr.send();
```

- 5. Recibe la respuesta

¿Cómo saber cuándo ha llegado una respuesta del servidor? Bueno, XHR tiene dos propiedades que se utilizan para indicar una respuesta del servidor. El primero es "readyState", y el segundo es "status".
La propiedad "readyState" registra cómo progresa la solicitud. Devuelve un valor numérico, numerado del 0 al 4, que indica diferentes estados de progreso. Los números se traducen de la siguiente manera:
0 - solicitud no inicializada
1 - conexión al servidor establecida
2 - solicitud recibida por el servidor
3 - el servidor está procesando la solicitud
4 - la solicitud ha sido procesada y la respuesta está lista

- La propiedad "estado" indica si la solicitud se ejecutó con éxito o no.
  200 - solicitud ejecutada con éxito y respuesta entregada
  404 - página no encontrada
  Puede acceder a estas propiedades haciendo referencia a ellas desde la variable XHR de la siguiente manera: xhr.readyState o xhr.status
- De hecho, es mejor probar estos valores en la función (mencionada en el punto 3 anterior) antes de intentar recuperar los otros valores. Antes de recuperar cualquiera de las otras variables del XHR (por ejemplo, el texto de respuesta), debemos asegurarnos de que "readyState" sea 4 y "status" sea 200.
- Por lo tanto, nuestro código de función debe reescribirse de la siguiente manera:

```
function () {
    if (xhr.readyState == 4 && xhr.status == 200) {
        document.getElementById("info").innerHTML = xhr.responseText;
    }
}
```

- Entonces, ¿cómo sabemos que ha llegado una respuesta del servidor? Bueno, XHR tiene un evento que se activa cada vez que cambia el "readyState". Este evento se llama evento "onreadystatechange". Este evento es la manera perfecta de ejecutar cualquier función diseñada para utilizar los resultados recuperados del servidor.
- Simplemente asigne la función para responder a este evento de la siguiente manera: onreadystatechange = function_name;
- Como estamos usando una función anónima, la adjuntamos de la siguiente manera:

```
xhr.onreadystatechange = function () {
    if (xhr.readyState == 4 && xhr.status == 200) {
        document.getElementById("info").innerHTML = xhr.responseText;
    }
}
```

- Hemos completado los cinco pasos y nuestra implementación de AJAX está lista para comenzar. Simplemente tenemos que reunir todo bajo una función llamada “cargarNuevoTexto ()" (el nombre que le asignamos anteriormente). Entonces, todo se une así:

```
document.getElementById('btn').addEventListener('click', cargarNuevoTexto);

function cargarNuevoTexto() {
    var xhr = new XMLHttpRequest();
    xhr.open("GET", "documento.txt", true);

    xhr.onreadystatechange = function () {
        if (xhr.readyState == 4 && xhr.status == 200) {
            document.getElementById("info").innerHTML = xhr.responseText;
        }
    }
    xhr.send();
}
```

Cuando usar Ajax

- Ajax se introdujo en un intento de hacer que las aplicaciones web tengan la sensación de los programas de escritorio en términos de tiempo de respuesta.
  AJAX ofrece una experiencia de usuario más rica al mantener al usuario en contexto. En lugar de esperar a que se cargue la página, el usuario accede a nuevo contenido en la misma página. Esto da una sensación de continuidad y consistencia.
  Ajax mejora el tiempo de carga. Esto se debe a que solo se cargan secciones de la página web. Esto hace que la página web aparezca más rápido.
  Utiliza menos ancho de banda. Cargar secciones de una página consume menos ancho de banda que cargar toda la página. Además, AJAX puede usar formatos de datos compactos como JSON. Es más pequeño en comparación con formatos de datos más descriptivos como XML o HTML. También reduce la carga del servidor.