JavaScript

Programa del curso

- Clase 1: Introducción y sintaxis
- Clase 2: Funciones, arrays y objetos
- Clase 3: Javascript integrado a HTML
- Clase 4: DOM, selectores y elementos
- Clase 5: Eventos
- Clase 6: Formularios
- **Clase 7: Ajax**
- Clase 8: Ejercicio integrador

1.
AJAX

JSON

Formularios - JSON

El formato de datos de JavaScript Object Notation, o JSON para abreviar, se deriva de los literales del lenguaje de programación JavaScript. Como un subconjunto del lenguaje JavaScript, no es un lenguaje de programación, sino un formato de intercambio de datos.

En su estructura es muy parecido a un objeto literal de JavaScript con algunas diferencias. Los nombres de las propiedades se escriben entre comillas dobles y los valores de string también.

```
var objetoEnFormatoJSON = '{ "atributo": "valor", "atributo1": 1, "atributo2": [], "atributo3": null, "atributo4": false }';
```

Formularios - JSON

Javascript tiene un objeto JSON que tiene los métodos stringify() y parse().

Stringify: El método stringify permite pasar un objeto o valor de JavaScript al formato JSON.

Parse: El método parse toma una cadena de caracteres en formato JSON y lo transforma en un objeto o valor de Javascript.

Gracias a estos 2 métodos se puede utilizar el formato JSON para intercambiar datos en formato de texto.

Formularios - JSON

Teniendo una variable en formato JSON se accede a sus atributos de la siguiente manera:

var objetoEnFormatoJSON = JSON.parse("atributo2": [{“numero”:1},{“numero”:2}]’);

objetoEnFormatoJSON.atributo2[1].numero

AJAX

AJAX

Ajax en sí no es una tecnología, sino más bien un término implementado por Jesse James Garrett en 2005.

Ajax significa Asynchronous JavaScript and XML (a.k.a. Ajax) y se ha convertido sinónimo del desarrollo moderno de front-end, y por una gran razón. Ofrece la posibilidad de iniciar peticiones HTTP como GET y POST a petición sin tener que navegar fuera de la página web actual.

Es por eso que podemos decir que Ajax es una técnica para crear webs dinámicas.

AJAX

AJAX nos permite:

- Refrescar el contenido de una página sin recargarla
- Pedirle información a un servidor
- Recibir información de un servidor
- Enviarle información a un servidor

AJAX - Pasos

1. var xmlhttp = new XMLHttpRequest();

AJAX - var xmlhttp = new XMLHttpRequest();

El único método global de la interfaz XMLHttpRequest es el del constructor, que, cuando se invoca, devolverá a nuestra aplicación una nueva instancia del objeto XMLHttpRequest. Será a través de la interfaz heredada por este objeto que vamos a iniciar y gestionar nuestras solicitudes.

Como la solicitud HTTP se produce asincrónicamente, es necesario que nuestra aplicación sea notificada de cualquier cambio de estado, durante la duración de la solicitud. Tales notificaciones pueden ser si la respuesta se ha cumplido o la conexión ha expirado.

De esta forma proporciona una forma fácil de obtener información de una URL sin tener que recargar la página completa. XMLHttpRequest es ampliamente usado en la programación AJAX.

AJAX - Pasos

1. var xmlhttp = new XMLHttpRequest();

2. xmlhttp.onreadystatechange = function() {
if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
console.log(xmlhttp.responseText);
//Mi código
}
};

xmlhttp.onreadystatechange

Una función del objeto JavaScript que se llama cuando el atributo readyState cambia. El callback se llama desde la interfaz del usuario.

xmlhttp.readyState

0 si no se inicializó, 1 si está cargando, 2 si ya se envió el pedido, 3 si esta descargando la respuesta y 4 si terminó

xmlhttp.status

El estado de la respuesta al pedido. Este es el código HTTPresult (por ejemplo, status es 200 por un pedido exitoso). Solo lectura.

xmlhttp.responseText

La respuesta al pedido como texto, o null si el pedido no fue exitoso o todavía no se envió. Solo lectura.

AJAX - Pasos

1. var xmlhttp = new XMLHttpRequest();

2. xmlhttp.onreadystatechange = function() {
if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
console.log(xmlhttp.responseText);
//Mi código
}
};

1. xmlhttp.open("GET", "url", true);

AJAX - xmlhttp.open("GET", "url", true);

Inicializa el pedido. Este método es para ser usado desde código JavaScript.

method
El método HTTP a usar: tanto "POST" o "GET". Se ignora para urls que no son de HTTP.

url
La URL a la que se envía el pedido.

async
Un parametro opcional, booleano que por defecto es true. Indica si la operación se realiza de manera asincrónica.

AJAX - Pasos

1. var xmlhttp = new XMLHttpRequest();

2. xmlhttp.onreadystatechange = function() {
if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
console.log(xmlhttp.responseText);
//Mi código
}
};

1. xmlhttp.open("GET", "url", true);
2. xmlhttp.send();

AJAX - xmlhttp.send();

Envía el pedido.

AJAX – Repasamos los Pasos

1. var xmlhttp = new XMLHttpRequest();

2. xmlhttp.onreadystatechange = function() {
if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
console.log(xmlhttp.responseText);
//Mi código
}
};

1. xmlhttp.open("GET", "url", true);
2. xmlhttp.send();

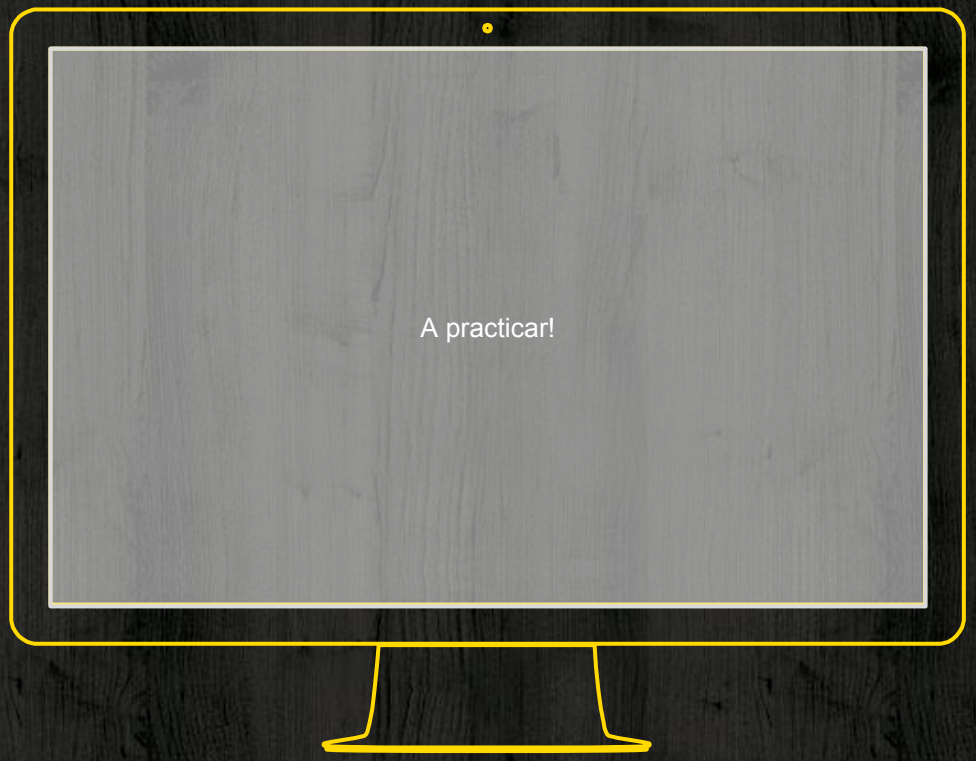
AJAX - Parametros

```
xhttp.open("POST", "ajax_info.txt", true);

xhttp.setRequestHeader("Content-type",
"application/x-www-form-urlencoded");

var params = “nombres=Nicolas&apellidos=Isnardi”

xhttp.send(params);
```



A practicar!

Práctica 7 – **JSON y AJAX**

Gracias!

Preguntas?