AJAX apunte

AJAX (*Asynchronous JavaScript and XML*) permite la **comunicación asíncrona** entre el navegador y el servidor sin recargar toda la página. Esto hace posible acciones dinámicas como recargar listas, guardar formularios o actualizar tablas de datos en segundo plano.

Cliente y Servidor

- **Cliente:** el navegador del usuario (yo y mi computadora)
- **Servidor:** computadora remota que responde solicitudes

El servidor procesa peticiones y devuelve respuestas. La comunicación se basa en el modelo **cliente-servidor**, donde el cliente inicia las solicitudes y el servidor responde.

¿Qué hace AJAX?

Permite que el navegador envíe/reciba datos del servidor sin tener que recargar toda la página.

AJAX es una combinación de:

- **HTML** y **DOM** para construir la página
- **Javascript** para enviar solicitudes
- `fetch()` (o `XMLHttpRequest` en versiones antiguas)
- **JSON** como formato de intercambio de datos

Estructura del ejemplo

Este código js se ejecuta cuando la ventana termina de cargar:

```
window.addEventListener("load", function() {
    log("Window loaded");
    // Se definen botones, formularios y eventos
});
```

Esto permite que el script acceda al DOM con seguridad una vez que esté cargado.

Envío de datos al servidor (AJAX POST)

Cuando el usuario llena un formulario con datos de un producto y da clic en **"Enviar"**, se genera un objeto `Product` y se envía al servidor por medio de `fetch()` con método `POST`:

```
async function addProduct(product){
    const url = "/add_product";

    const response = await fetch(url, {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify(product)
    });

    if (response.ok) {
        const data = await response.json();
        log("Product added successfully:", data);
    } else {
        alert("Error adding product: " + response.statusText);
    }
}
```

Este fragmento envía datos al servidor sin recargar la página.

Recepción de datos del servidor (AJAX GET)

Cada vez que se solicita ver los productos disponibles, se ejecuta:

```
async function loadProducts(){
    let response = await fetch("/products");

    if (response.ok){
        let json = await response.json();
        const codeResult = document.getElementById("codeResult");
        codeResult.innerHTML = JSON.stringify(json.products);
    } else {
        alert("HTTP-Error:" + response.status);
    }
}
```

Este método obtiene datos de `/products` y los muestra en el DOM sin recargar.

Validación previa con js

Antes de enviar, los datos del formulario se validan para asegurarse de que no estén vacíos:

```
function generateProduct(){
    const idValue = myForm.elements["id"].value;
    const nameValue = myForm.elements["name"].value;
    const priceValue = myForm.elements["price"].value;

    let msg = "Created Product";
    if (!idValue) msg = "Id is empty";
    if (!nameValue) msg = "Name is empty";
    if (!priceValue) msg = "Price is empty";

    let newProduct = null;
    if (msg === "Created Product") {
        newProduct = new Product(idValue, nameValue, priceValue);
    }

    return { product: newProduct, msg: msg };
}
```

Refresco automático de datos

Cada 5 segundos, el sistema actualiza la lista de productos:

```
setInterval(() => {
    loadProducts();
}, 5000);
```

Esto mantiene la vista sincronizada con el servidor sin necesidad de intervención del usuario.

Integración con Grid.js

Para mostrar los datos de forma tabular y ordenada, se utiliza Grid.js:

```
gridTable = new gridjs.Grid({
    columns: ["Id", "Name", "Price"],
    pagination: true,
    search: true,
    sort: true,
    server: {
        url: "/products",
        then: data => data.products
    }
}).render(wrapper);
```

Grid.js se encarga de renderizar la tabla, manejar la paginación, búsqueda y ordenamiento, solicitando datos dinámicamente desde el servidor.

Entonces,

- Se envían productos con `POST`
- Se cargan productos con `GET`
- Se evita la recarga de página
- Se muestran los datos dinámicamente con Grid.js