***AJAX :***

***Une définition globale :***

AJAX signifie :

Asynchronous

JavaScript

And

XML.

Ce n’est pas un langage informatique mais une technologie qui permet de faire appel à un serveur depuis une page web. Au lieu de refresh toute une page, l’AJAX permet de le faire que sur une partie de celle-ci pour économiser des ressources et améliorer l’expérience utilisateur.

***Sans serveur, seul Firefox permet de faire des tests d’AJAX.***

Un exemple concret :

<script>

function loadDoc(param1, param2) { //Le paramètre 1 cible le fichier texte, le paramètre 2 le conteneur où on veut envoyer.

var xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function () {

if (this.readyState == 4 && this.status == 200) { // si readystate =4 et this.status =200 alors tout est OK.

document.querySelectorAll(param2).innerHTML =// Ici, queryselector permet de cibler vraiment ce qu'on veut.

this.responseText; // intègre le texte à la div ciblée.

}

};

xhttp.open("GET", param1, true);//"GET" permet de dire au serveur de lui prendre quelque chose, et true signifie que tout

//s'exécute de façon asynchrone et qu'on réccupère la main tout de suite pendant qu'il traite la demande.

xhttp.send(); // Permet d'envoyer la requête au serveur.

}

</script>

La fonction type à reproduire :

function loadDoc(x,y,z) {

var xhttp = new XMLHttpRequest();

console.log(xhttp);

xhttp.onreadystatechange = function() {

if (this.readyState == 4 && this.status == 200) {

document.querySelectorAll(y)[z].innerHTML =

this.responseText;

}

};

xhttp.open("GET", x, true);

console.log(xhttp);

xhttp.send();

}

function choix1(){

loadDoc("texte1.txt",".div",0);

}

function choix2(){

loadDoc("texte2.txt",".div",1);

}

ReadyState :

Le ready State est l’état d’avancement du dialogue entre le serveur et la page web.

Il est découpé en 5 phases de 0 à 4.

0 : création de la requête

1 :On établit la connexion avec le serveur.

2 : La réponse du serveur disant qu’il a reçu la requête.

3 : Le serveur dit qu’il traite la requête.

4 : La requête est finie, le serveur l’a envoyée.

En général, les devs ne testent que le 4 pour ne traiter que les réponses positives :

if (this.readyState == 4 && this.status == 200) {

document.querySelectorAll(y)[0].innerHTML =

this.responseText;

}

Mais il est également possible de traiter les autres cas en car d’erreurs pour prévenir l’utilisateur (ou le dev via la console) s’il y a un problème :

if (this.readyState != 4) {

console.log("Pas de reponse encore (" + this.readyState + ")");

}

if (this.readyState == 4) {

document.getElementById("pdemo").innerHTML = this.responseText;

console.log("Reponse arrivee.");

}

Les status du serveur :

Le serveur a plusieurs codes pour prévenir de son statut :

Les codes 200 : Tout va bien.

Les codes 300 : Il y a un changement au niveau du serveur, nom de domaine, ect, mais ce n’est pas une erreur.

Les erreurs 400 : Elles témoignent d’une erreur (erreur 404 fichier non trouvé) ou 403 pour dire qu’on n’a pas l’accès par exemple.

Les erreurs 500 : C’est une erreur mais côté serveur (ex : il y a trop de demandes et le serveur a planté).

L’exemple du menu select pour changer de langues mais avec Ajax :

function loadDoc(x, y) {

var xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function () {

if (this.readyState == 4 && this.status == 200) {

document.querySelectorAll(y)[0].innerHTML =

this.responseText;

}

};

xhttp.open("GET", x, true);

xhttp.send();

}

var choix = document.getElementById("choix");

function language() {

if (choix.value == "fr") {

loadDoc("titrefr.txt", "h2");

loadDoc("textefr.txt", "div");

}

else if (choix.value == "en") {

loadDoc("titreen.txt", "h2");

loadDoc("texteen.txt", "div");

}

else if (choix.value == "es") {

loadDoc("titrees.txt", "h2");

loadDoc("textees.txt", "div");

}

}

Dans la pratique, beaucoup mettent un nom d’ID à chaque div pour pouvoir les cibler beaucoup plus facilement qu’en les comptant manuellement, puisque les erreurs peuvent être fréquentes en agissant de la sorte.