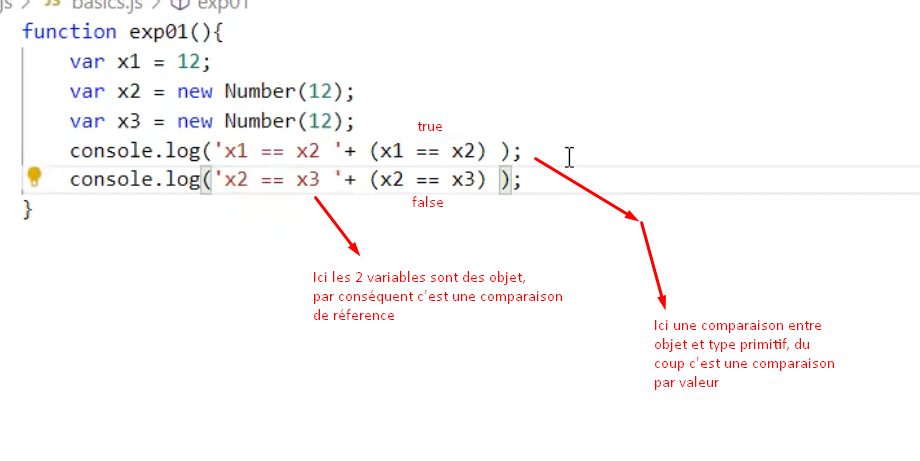
Types primitifs en JavaScript : number, string, boolean.

Les variables déclarer en d’hors de tous blocks se sont considérés comme des objets globals et il appartient à l’objet window, ils sont déclarés avec le mot clé var 🡪 var x = 3 ; x est accessible aussi par window.x ;

Alert(‘ici un message’) 🡪 permet d’afficher une fenêtre de notification.

Confirm(‘répondez avec Oui, pour confirmer’) 🡪 affiche un message de confirmation, il retourne un boolean.

Window.setInterval(function, mills) 🡪 permet d’appeler périodiquement une fonction.



Le triple === fait une comparaison de contenu et de type.

Quelque fonction sur les chaines de caractères :

s.indexOf(O) 🡪 retourne l’index de O dans la chaine s.

s.includes(‘’) 🡪 équivalent á contains en java 🡪 permet de chercher une chaine dans une chaine.

s.lenght 🡪 retourne la taille de la chaine, il s’agit d’un attribut, par contre dans java c’est une méthode.

s.substring(début, fin) 🡪 retourne la sous chaine contenant entre début et fin - 1.

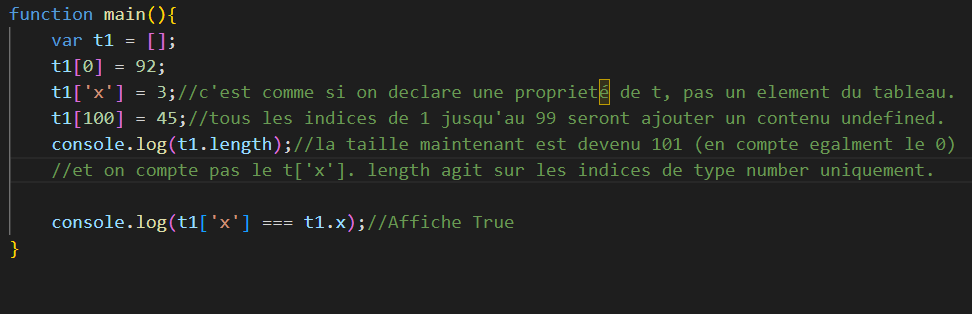
s.charAt(7) 🡪 permet de retourner le caractère qui est á l’endroit mentionné.

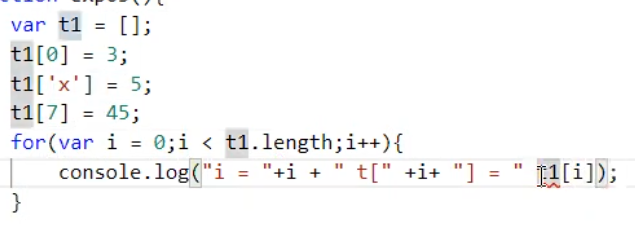
Les tableau en JavaScript :

Var t1 = [] 🡪 déclaration du tableau vide.

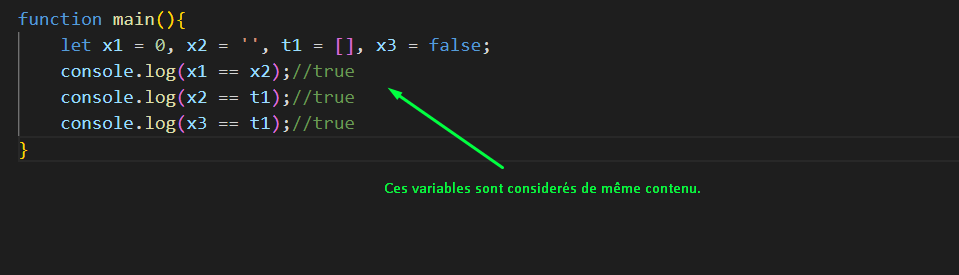
Var t2 = new Array() 🡪 instanciation avec Array, mais pas de différence parraport á t1 = [].

Var t3 = [1 , ‘T’, ‘’coco’’, null, true] 🡪 un tableau en JavaScript peut contenir plus qu’un type á la fois.

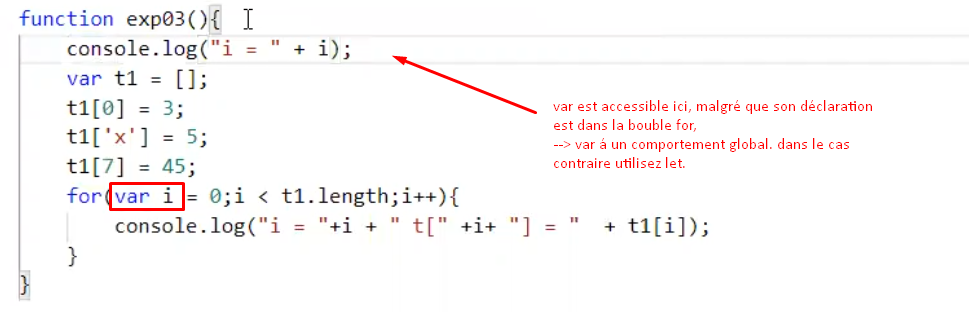




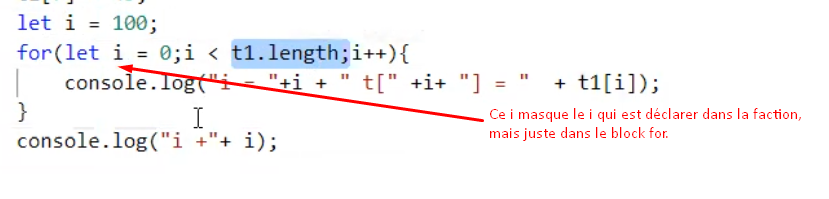
Cette bouble affiche toutes les élements du tableau, même les undefind, mais non pas les proprietés du tableau, tel que x.

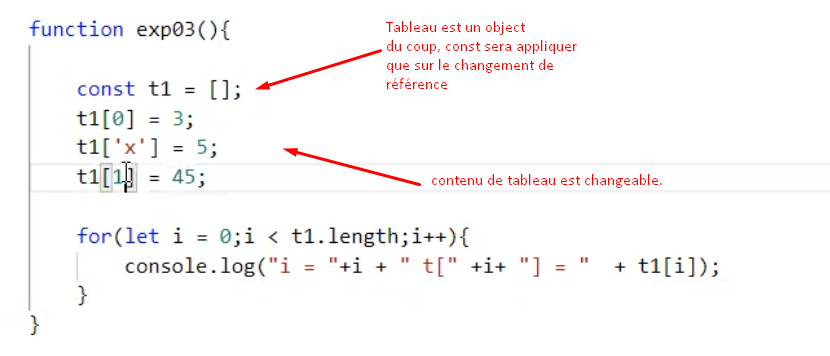


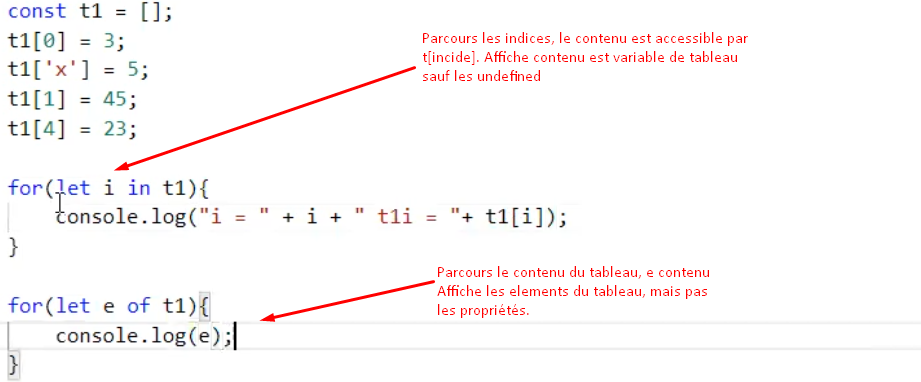
Diffèrence entre let et var.



Ce code même en utilisant let, génera une erreur d’interprétation (i not defined). Pour les variables déclarer var, sont accessible dés la première intruction de la fonction .





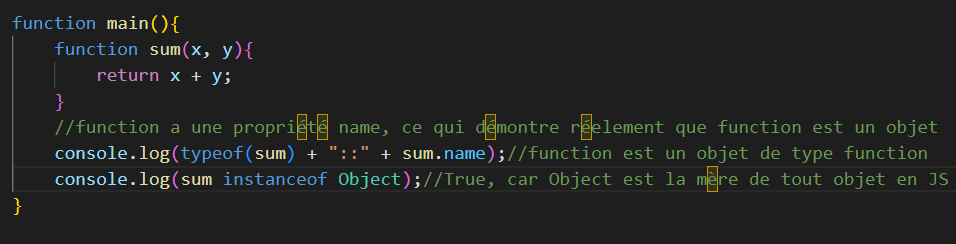


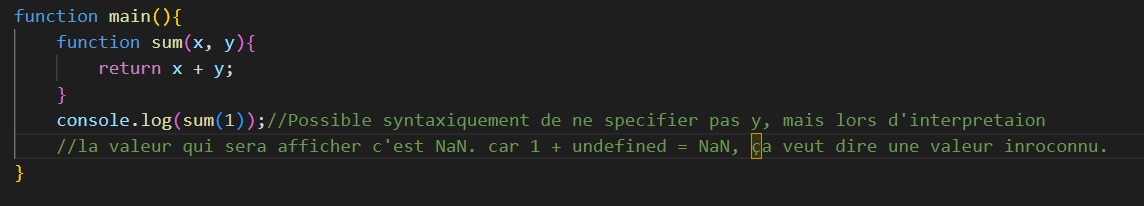
Il y’a aussi la boucle t.forEach( e => console.log(e)) qui parcours les élements du tableau et pas les indices, n’affiche ni les undefined, ni les proprièté du tableau.

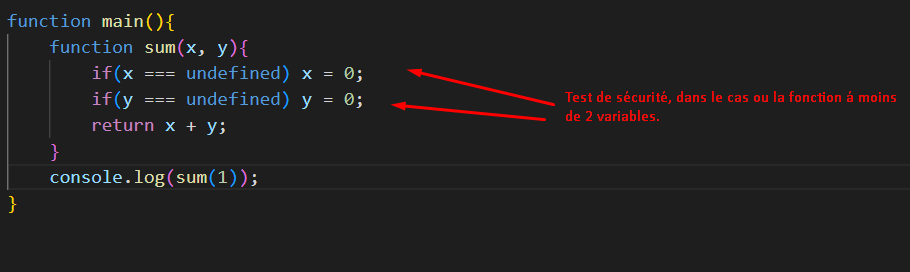
t.forEach( (e, i ) => console.log( i + ‘’ :: ‘’+ e) 🡪 permet d’afficher aussi les indices.

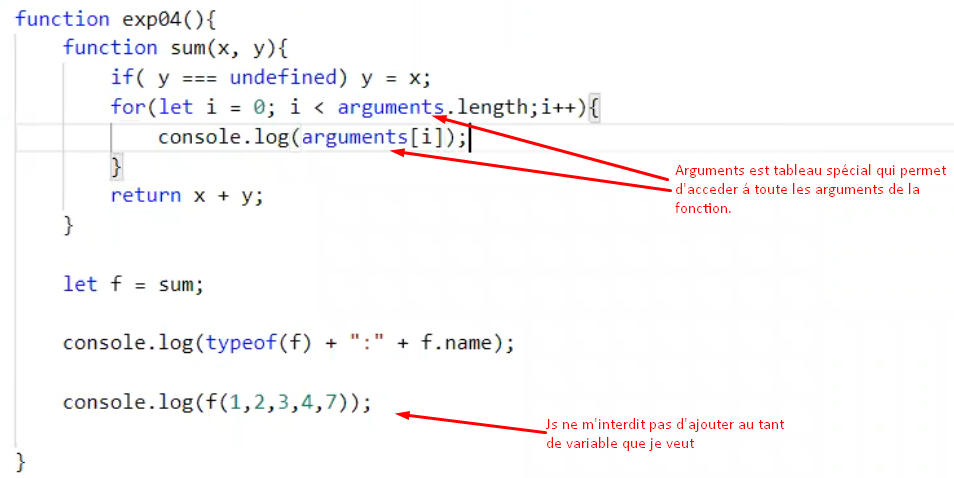
Donc la boucle pour afficher touts sauf les undefined, et que moi je trouve la meilleure c’est for(let i in t).

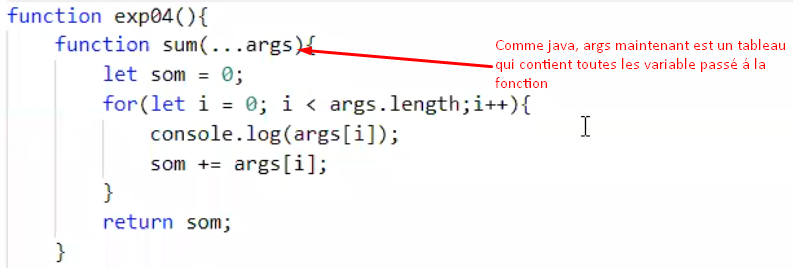
Fonction en JavaScript





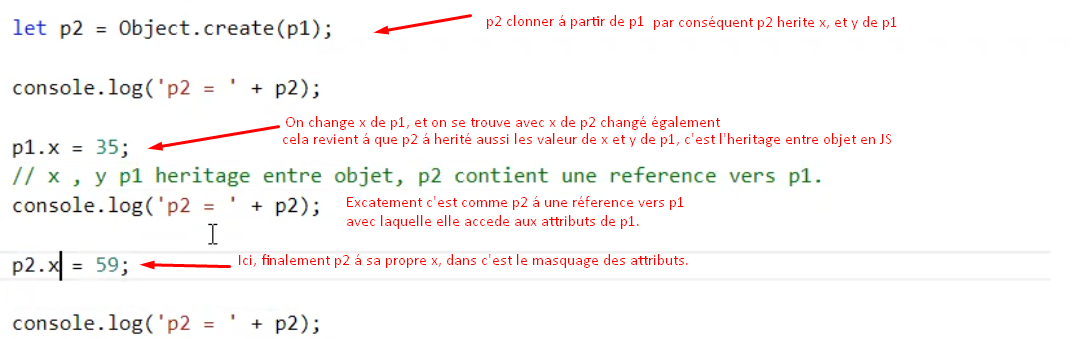
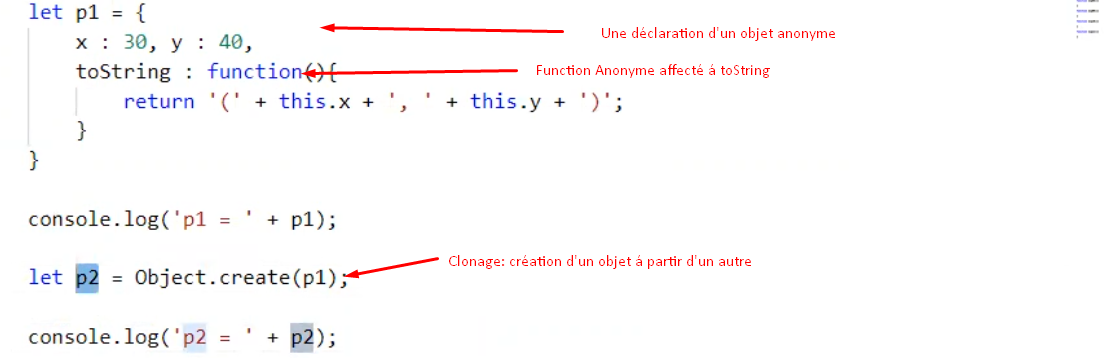




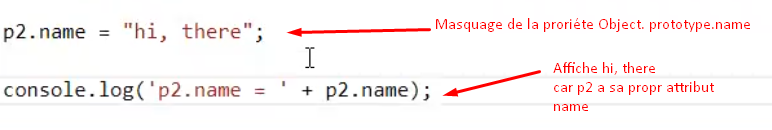


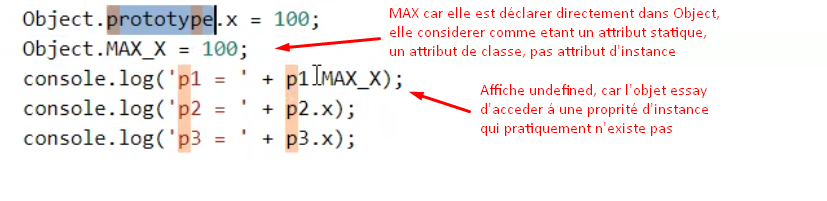
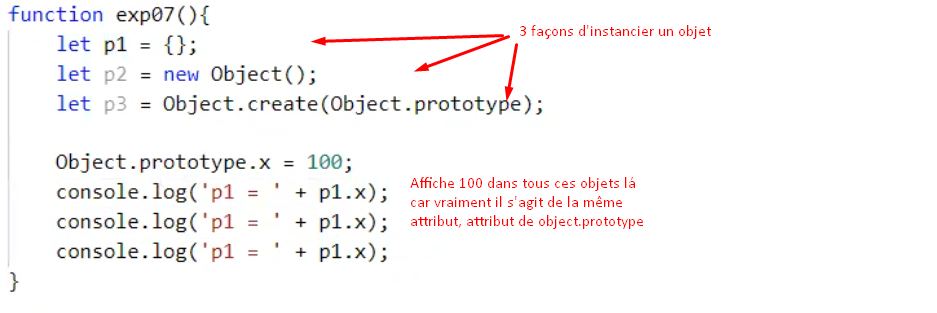
Args est un vrais tableau, par contre arguments ressemble á un tableau.

Les Classes en JavaScript.

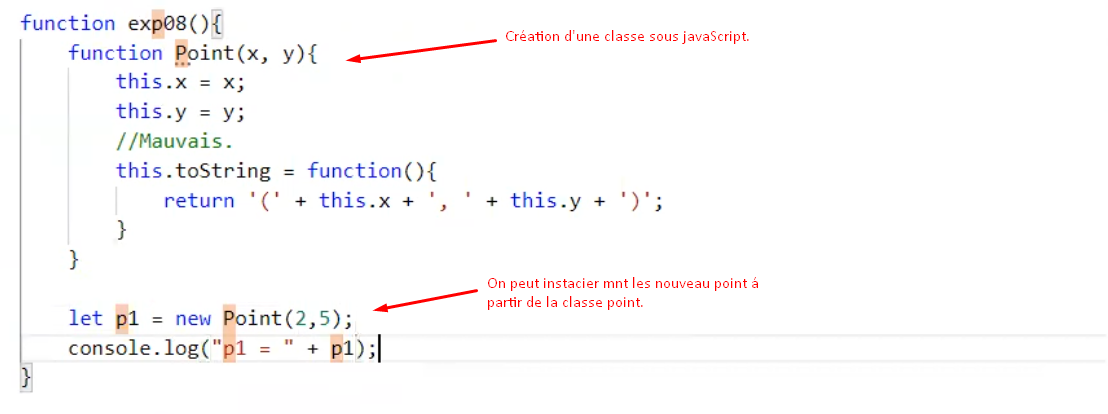


Tout objet en JS a une référence vers le prototype de son classe, et ce prototype de classe a egalement une reference vers le prototype de la classe object. En conséquence tout objet a une reference vers le prototype object.

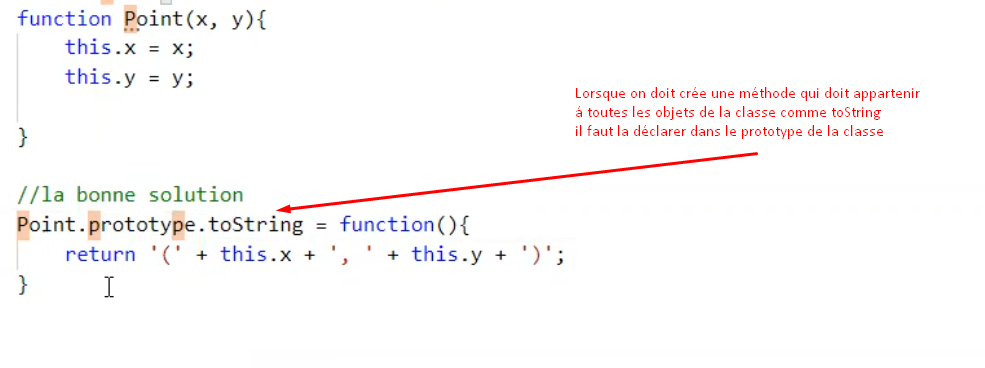




Création d’un classe á l’aide d’une fonction JS :

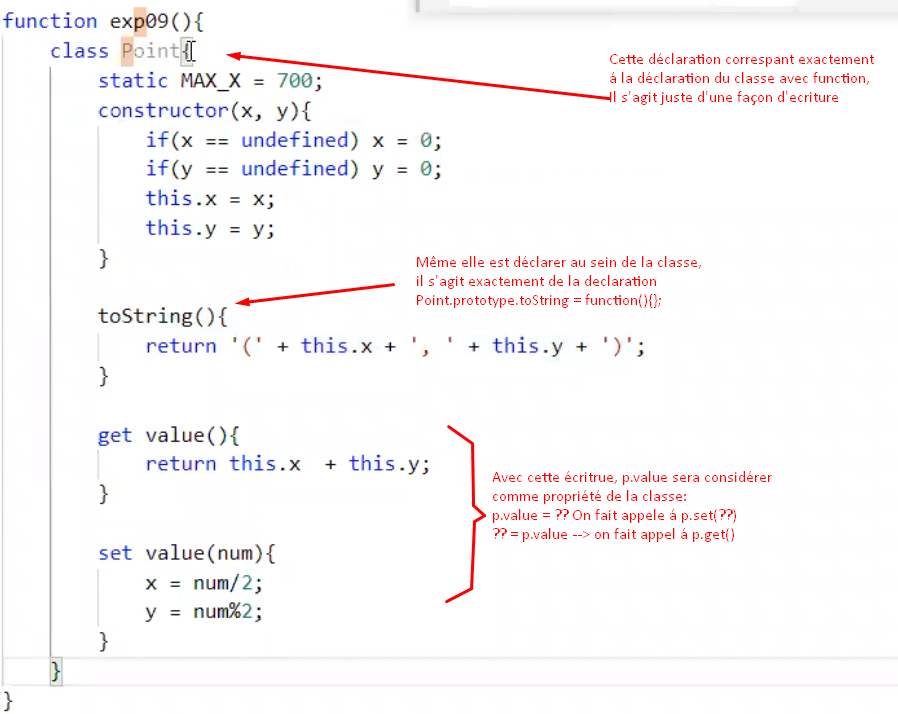


La définition des méthodes :



Donc pour optimiser la mémoire c’est mieux de déclarer les méthode de la classe á l’aide de Class.prototype.methodName = function(){} ;

La création de la classe á l’aide du mot clé class



**Technologie AJAX:**

**AJAX** est une technologie utilisé pour l'envoie d'une requete au serveur d'un manière asynchrone, c’est-à-dire que le client envoie une requete au client sans resté bloqué en attendant la réponse, grace á ajax le client peut continuer de naviguer sur la page pendant que la requete et la reponse sont tourné en arriere plan.

**XMLHttpRequest** 🡪 est l'objet utilisé pour envoyé des requetes asynchonre.

**XMLHttpRequest.readyState** == 4 🡪 veut dire que le serveur á recu la requete et il a génerer une réponse pour vous.

**XMLHttpRequest.status** == 200 🡪 ici lorsque le serveur á arriver au state 4, c’est-à-dire qu'il a une réponse , on test est ce que la réponse s'agit d'une valide réponse, c’est-à-dire la ressource demandé existe.

**XMLHttpRequest.responseText** 🡪 contient la réponse du serveur sous format textuelle.

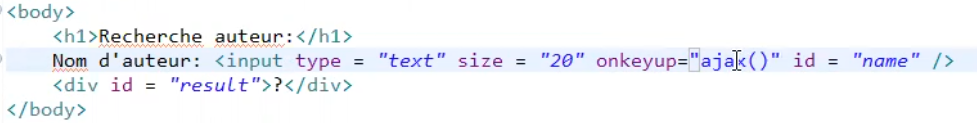
**XMLHttpRequest.onreadyStateChange** 🡪 recoit une fonction callback qui va etre executer á l'evenement de changement de state de serveur.

**XMLHttpRequest.open('GET', URL, true**) 🡪 définition de la requete asynchrone par la méthode http et l'url de serveur, true veut dire activé le mode asynchrone.

**XMLHttpRequest.send()** 🡪 justement pour envoyer la requete lorsqu'elle est ouverte.



L'appelle á la méthode ajax au sein d'une page HTML: dés qu'on laisse un touche du clavier, la fonction ajax sera invoqué, et la réponse sera disponible dans le div d'id result.



**La version finale de AJAX:**

L'idée est d'encapsuler les méthodes ajax dans une classe nommé AJAX (Single responsibilité) afin de la réutiliser dans plusieurs projet.



La réutilisation de la classe ajax va se faire par une instantiation de l'objet Ajax dans la classe et d'apple directement les méthodes GET OU POST:

