Connaissances générales

Closure: fonctions qui ont accès à des variables qui ne sont pas définies à l'intérieur de la fonction elle-même, mais plutôt dans la portée parente. En d'autres termes, une closure est une fonction qui "capture" les variables de son environnement lexical.

Scope: étendue d’une variable, c’est-à-dire l’endroit où elle est accessible dans le code

Exemple Closures en JavaScript:

```
1   //@Ibrahima Gabar Diop
2   //Illustration de la notion de Closure en Java
3   function multiplication(n) {
4     //fonction anonyme locale déclarée à l'intérieur de la fonction multiplication()
5     return function(x) {
6       return x * n;
7     }
8   }
9
10  //On déclare une variable doubler, à laquelle on affecte la fonction multiplication() qui prend deux
    comme attribut, par défaut.
11  var doubler = multiplication(2);
12  //On affiche le résultat de l'opération qui consiste à doubler la valeur de 5. Ici la fonction doubler
    prend en paramètre 5 et multiplie cette valeur par 2 ( ici x=5 et n=2)
13  console.log(doubler(5)); // affiche 10
14
15  //On affiche le résultat de l'opération qui consiste à tripler la valeur de 5. Ici la fonction tripler
    prend en paramètre 5 et multiplie cette valeur par 3 ( ici x=5 et n=3)
16  var tripler = multiplication(3);
17  console.log(tripler(5)); // affiche 15
18
```

Dans cet exemple, la fonction multiplication() renvoie une autre fonction qui utilise la variable n. Cette fonction interne peut être appelée en dehors de la fonction multiplier(), et elle se souvient de la valeur de n. Ainsi, la variable doubler est définie comme une fonction qui multiplie un nombre par 2, tandis que la variable tripler est définie comme une fonction qui multiplie un nombre par 3. Lorsque vous appelez ces fonctions avec un nombre donné, elles retournent le résultat de la multiplication de ce nombre par 2 ou 3, respectivement.

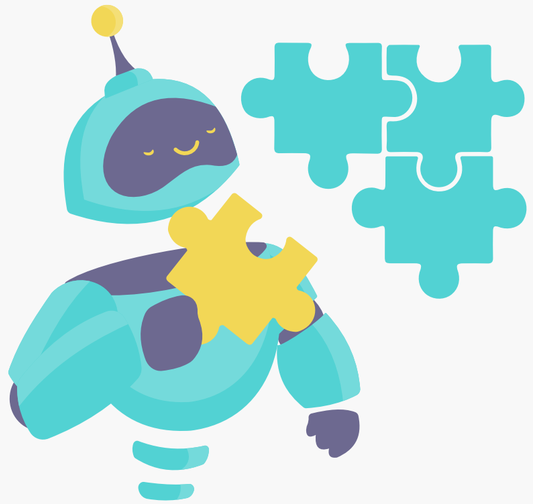
Exemple Scope en JavaScript:

```
var x = 10; // On déclare d'abord une variable globale x

//Ensuite on utilise une fonction où on déclare une variable y qui sera une variable locale
function Gabar() {
  var y = 20; // variable locale
  console.log(x); // affiche 10
  console.log(y); // affiche 20
}

Gabar();
console.log(x); // affiche 10
console.log(y); // Erreur : y n'est pas défini
// La variable y n'est accessible qu'à l'intérieur de la fonction Gabar()
```

Dans cet exemple, x est une variable globale, ce qui signifie qu'elle peut être utilisée dans toutes les parties du code JavaScript. y, quant à elle, est une variable locale qui n'est accessible qu'à l'intérieur de la fonction Gabar().



Conclusion:

En résumé, les Scopes et les Closures en JavaScript sont des concepts clés que vous devez comprendre pour écrire du code JavaScript efficace.