Documentation d'utilisation de la calculatrice :

La calculatrice JavaScript que j'ai développé est une fonction JavaScript prenant en argument une expression mathématiques correcte en chaine de caractères, c'est-à-dire une expression mathématique qui se trouve entre deux guillemets (double ou simple cote mais par convention il est préférable d'utiliser les doubles cotes car les simples cotes représentent un caractère dans d'autres langages informatiques. Toutefois le JavaScript étant très permissif vous pouvez utiliser les doubles ou les simples cotes.) Ainsi ma calculatrice fonctionne grâce à une fonction que j'ai appelé sombrement calcule prenant une string (votre expression mathématiques) et vous retournant une nouvelle string qui sera la réponse de votre calcul. Pour gérer votre expression mathématiques ma fonction calcul va s'aider de nombreuses regex (expression réguliers) qui sont des instructions permettant de repérer des mots ou symboles bien précis dans une chaine de caractères. Les regex me permettent donc de repérer les différents opérateurs présents dans une expression afin de pouvoir effectuer la bonne opération lors de la rencontre avec celui-ci. Mais aussi de repérer les parenthèses et les opérations se trouvant à l'intérieurs de ces dernières afin de pouvoir garder la priorité des parenthèses aux dessus de celles des opérateurs.

Pour utiliser ma calculatrice il vous suffira d'écrire
console.log(calcule(«votre_expression_mathematique »))

(console.log vous permettra de voir votre résultat dans votre terminal)

à la fin du fichier calculatrice.js et de lancer le fichier avec la commande node ./calculatrice.js.

Ensuite pour ce qui est de l'expression vous devez toujours mettre une expression mathématiques correctes entres guillemet (et de même type) et ne comportant aucun espace ainsi vous ne pouvez pas faire :

- calcule(2+5) : car il n'y pas de guillemets
- calcule(« 2+5') : car les guillemets ne sont pas du même type
- calcule(« 2++ ») : car l'expression n'est pas correcte
- calcule(« 2 + 5 ») : car comporte des espaces

Vous pourrez donc faire des additions, multiplication, soustractions et divisions avec les symboles suivants :

- + : ex calcule(«2+5 ») , calcule(« (2+3)+5 »)
- * : ex calcule(«2*5 »), calcule(« (2*3)*5 ») , calcule(«(+2*-3)*-5 »)
- - : ex calcule(« 2-5 »), calcule(« (2-3)-5) »), calcule(«(+2--3)-+5 ») (Attention le double '-' n'est utilisable que dans la soustraction ou dans un cas de mathématiques correct si autre cas il existe. Donc ne faites pas 2*--3 => incorrect)
- / : ex calcule(« 2/5 »), calcule(« (2/-5))/5 »)

Vous pouvez aussi faire des pourcentages avec le symbole % (qui n'est pas un modulo ici attention) , des puissances (et par extension des carrées) avec le symbole ^ et des racines carrées avec la syntaxe sqrt(« expression »)

- % : calcule(« 10%100 ») => permet de calculer 10 % de 100 , ex calcule(« 25%95 ») => 25 % de 95
- ^ : calcule(« 2^2 ») => carrée (ou puissance 2) de 2, calcule(« 2^18 ») 2 puissance 18
- sqrt : calcule(« sqrt(2) ») => racine carrée de 2, calcule(«sqrt(16 ») => racine carrée de 16.

Enfin sachant que la calculatrice gere les entiers flottants. Pour les utiliser il suffit de mettre un point , exemple :

- calcule(1.5+2.5),
- calcule(0.5*4),
- calcule(5.5-3.75),
- calcule(10.75/6.24)

Voici comment procède ma calculatrice lorsque vous entrez un calcul en paramètre :

(**Explication du code dans les grandes lignes**)

- Tout d'abord elle vérifie, grâce à une regex, que votre expression n'est pas composé que d'un chiffre ou nombres avec des ou sans parenthèses : ex « 5 » « (-5) » ou encore « +5 » car dans ce cas-là il n'y a pas d'opérations et donc pas de calcul à faire ainsi elle vous retourne directement ce que vous avez donné.
  Dans le cas inverse ou il y aurait une ou plusieurs opérations alors la fonction va rentrer dans une boucle while qui ne se finira que si la regex ne trouve qu'un seul nombre ou chiffre dans l'expression

- Une fois dans cette boucle while nous allons utiliser plusieurs regex qui auront comme utilisé de trouver les racines carrées avec ou sans parenthèses, puis trouver les multiplications, divisions, puissance et pourcentage (avec ou sans parenthèses) et enfin les additions et soustractions (avec ou sans parenthèses). Ces trois blocs dans cet ordre vont permettre à notre calculatrice de respecter la priorité de calcul en fonction de l'importance de l'opérateur. Une fois rentré dans un de ces blocs le programme va alors extraire une sous chaine de caractères avec toujours la forme suivante « a operateur b) ou a et b sont des nombres ou chiffres. Grace à cette sous expression il est alors possible de déterminer plus en détails l'opérateur parmi les operateurs pris dans un des trois blocs d'opérateurs. Une fois dans le bon cas alors le programme effectuera le calcul est mettra le résultat dans l'expression originel à la place de la sous expression prise un peu plutôt. Cela permettra de réduire boucle après boucle l'expression en fonction de sa longueur et de sa complexité

- Enfin une fois l'un de ces blocs effectués lors de votre itération de boucle la fonction va revérifier s'il reste un nombre dans l'expression ou non. Il va aussi vérifier si dans un cas précis l'expression mathématiques ne serait pas devenue « Infinity » (valeur Infinie) qui peut être obtenu lors de calcul bien précis comme effectué une division par 0. Dans ce cas le programme vous renverra la valeur Infinity et forcera l'arrêt de la boucle et du programme.