Dans la suite du document, si vous pensez avoir la bonne intuition (réponse), vous mangez une 🍒[1]

Au final, comptez le nombre de cerises mangées, cela me permettra d'évaluer votre gourmandise !

Question 1

Un bloc conteneur (en noir) contient deux blocs, it1 (en rouge) et it2 (en bleu).

Pouvez-vous modifier la figure suivante si le bloc conteneur à la propriété display:flex ?

Question 2

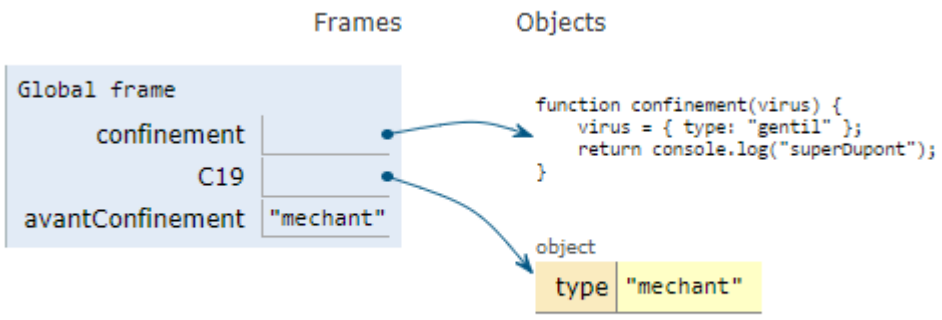
Soit le code JS suivant :

```
1. function confinement(virus) {
2.     virus = { type: "gentil" };
3.     return console.log("superDupont");
4. }
5.
```

[1] une cerise !

```
6. let C19 = { type: "mechant" };
7.
8.
9. let avantConfinement = C19.type;
10.confinement(C19);
11.let apresConfinement = C19.type;
```

La figure suivante représente l'état de la mémoire juste avant l'appel de la fonction confinement en lig.10



Pouvez-vous me donner étape par étape le comportement en mémoire de l'exécution et la valeur finale de la variable "apresConfinement".

Question 3

Soit un tableau de virus :

```
const virus = ["Arterivirus", "Cytomegalovirus", "H1N1", "Lambda phage", "Myxovirus parotitidis", "H3N2", "Lentivirus", "Leporipoxvirus", "Astrovirus", "Langur virus", "Ateline herpesvirus group",

 "Cytomégalovirus", "Rabbit fibroma virus", "Cytomegalovirus", "Cytomegalovirus", "Vaccinia virus", "SUPERDUPONT", "Vacuolating virus", "Coronavirus"];
```

Que vaut le résultat de la méthode :

```
virus.reduce((a, x) => {

        if(!a[x[0]]) a[x[0]] = [];

        a[x[0]].push(x);

        return a; }, {}

        );
```

Question 4

Que fait ce code ?

```
1. Function.prototype.bind = function(){
2.   var fn = this,
3.   args = Array.prototype.slice.call(arguments),
4.   object = args.shift();
5.   return function(){
6.     return fn.apply(object,
7.       args.concat(Array.prototype.slice.call(arguments)));
8.   };
9. };
```

Question 5

Comprenez l'intégralité de ce code.

```
1. function garde(n, iterable) {
2.
3.   const iter = iterable[Symbol.iterator]();
4.   return {
5.     [Symbol.iterator]() {
6.       return this;
7.     },
8.     next() {
9.       if (n > 0) {
10.        n--;
11.        return iter.next();
12.      } else {
13.        return {
14.          done: true
15.        };
16.      }
17.    }
18.  };
19.}
```

Notez dans le questionnaire (dans le blog) le nombre de cerises obtenues [0-4+] !