DESAFIO N 2 NODEJS

RODRIGO ECHARREN

1. Defina con sus palabras las diferencias que tienen entre las variables de JavaScript (var, let y const).

Las variables “var” y “let” pueden ser inicializadas o no en el momento de su declaración, y su valor puede ser modificado posteriormente indefinidas veces. Por el contrario, “const” tiene que estar inicializada con un valor al momento de su declaración, y éste no puede ser modificado posteriormente, ya que es constante.

Por otra parte, “var” puede ser declarada varias veces con distintos valores, lo que podría ocasionar posibles errores, en cambio “let” solo se permite declarar solo una vez, lo mismo que con “const” lógicamente.

Otra diferencia entre “var” y “let” es el scoope o alcance que tienen. “var” tiene un alcance global, es decir que su valor se podrá ir modificando de manera global sin importar el bloque en que se haga, siempre que estos estén en la misma función. En cambio “let” tiene alcance solo en el bloque en que fue creado.

2. Defina con sus palabras que entiende por event-loop de nodejs.

El event loop es un proceso que está constantemente en bucle revisando si hay operaciones qué ejecutar, a medida que se va encontrando con estas las va ejecutando o apilandoen en una cola para su posterior ejecución. Estas operaciones pueden ser bloqueantes (por ej. Console.log()), es decir que se deben ejecutar para dar paso a las siguientes y bloquean la escucha del event loop en todo el código. También existen las No

bloqueantes (por ej. setTimeOut()), que se coloca en la cola para su posterior ejecución pero no bloquean la escucha del loop en todo el código.

3. Que diferencia existe entre ejecución de tareas sincronías y asíncronas en el marco de nodejs.

Las tareas sincronicas (por ej. Console.log()) se ejecutan una después de la otra, son bloqueantes y lo hacen antes que las asincrónicas , ya que estas ultimas (por ej. setTimeOut()) son apiladas en una cola para su posterior ejecución y no bloquean la escucha del loop.

4. ¿Porque decimos que nodejs es de hilo único (single-thread)?

Porque a medida que el event loop va detectando distintas operaciones asincrónicas las va apilando en una cola fuera del hilo principal, para su posterior ejecución. De ésta manera puede seguir “escuchando” los distintos llamados o request que sucedan en ese único hilo sin que estas operaciones asincrónicas, que pueden durar tiempos elevados, lo bloqueen.

5. Realice dos funciones similar a la vista en clase donde exista una función sincrona y una función asíncrona y que estas tengan por lo menos 2 objetos globales.

```
6.    (() => {
7.        console.log("Inicio programa");
8.
9.        function uno() {
10.         console.log("sincronico 1 --> "+__dirname);
11.         dos();
12.         console.log("sincronico 2 --> "+__filename);
13.       }
14.
15.       function dos() {
```

```
16.         setTimeout(()=>{
17.           console.log("Asincronico 1 --> "+__dirname);
18.           console.log("Aincronico 2 --> "+__filename);
19.         },1000);
20.       }
21.
22.       uno();
23.       console.log("Fin Programa");
24.       console.log("----------------");
25.     })();
```

RESULTADO DEL PUNTO 5

```
PS C:\Users\rodry\Documents\CURSO NODE JS\Nueva carpeta> node appRodri2.js
Inicio programa
sincronico 1 --> C:\Users\rodry\Documents\CURSO NODE JS\Nueva carpeta
sincronico 2 --> C:\Users\rodry\Documents\CURSO NODE JS\Nueva carpeta\appRodri2.js
Fin Programa
----------------
Asincronico 1 --> C:\Users\rodry\Documents\CURSO NODE JS\Nueva carpeta
Aincronico 2 --> C:\Users\rodry\Documents\CURSO NODE JS\Nueva carpeta\appRodri2.js
PS C:\Users\rodry\Documents\CURSO NODE JS\Nueva carpeta>
```