CURSO: CC3S2

Tarea L-05 : Javascript -2.

Subir un archivo (CC3S2-LEC05-NOMBRE-APELLIDO.DOC) con los ejercicios resueltos.
Para cada pregunta: Presente la respuesta y muestre el resultado.

Las Funciones CallBack – se pasan como parámetros a funciones de nivel superior.

1. escribir una función llamada **each** que acepte dos parámetros: una matriz y una función de **callback**. La funcion **each** debe recorrer la matriz que se le ha pasado y ejecutar la función de **callback** en cada elemento. Ejecute la funcion each con tres funciones de callback diferentes.

Por ejemplo asi:

```
each([1,2,3,4], function(val){
    console.log(val);
}); //imprime cada elemento del arreglo
each([1,2,3,4], function(val){
    console.log(val*2);
}); //imprime cada elemento del arreglo multiplicado por 2

function each(){
    // Aqui va su código
}
```

2. Escriba una función llamada reject que acepte dos parámetros: una matriz y una funcion callback. La función map debe devolver una nueva matriz con todos los valores de la matriz que no retornan verdadero cuando se pasan a la función callback. P. Ej:

```
reject([1,2,3,4], function(val){
    return val > 2;
}); // [1,2]

reject([2,3,4,5], function(val){
    return val % 2 === 0;
}); // [3,5]
```

Utilizando setTimeout + setInterval
Es bastante común escribir código que queremos que se ejecute después de un período de tiempo específico. Tal vez sea necesario poner un temporizador en un juego o realizar alguna animación en la página después de que haya transcurrido un tiempo fijo. Para hacer esto, usamos las funciones **setTimeout y setInterval.** Ambas funciones aceptan una función de callback y un tiempo en milisegundos como parámetros. La principal diferencia es que la función **setTimeout** solo ejecutará la función de callback para que se ejecute una vez, mientras que **setInterval** la ejecutará una cantidad infinita de veces (hasta que se borre el temporizador).

setTimeout y setInterval devuelven un valor especial llamado identificador de temporizador. Si pasamos este valor al método **clearTimeout** o **clearInterval,** podemos detener nuestro temporizador

Que es lo que hace el siguiente código?

```
let timerId = setInterval(function(){
    console.log("HI!");
},1000);

setTimeout(function(){
   clearTimeout(timerId);
},3000);
```

Accediendo al DOM con Javascript

3. En el siguiente Código html: escribir ex03a.js, ex03b.js ,etc. Según lo solicitado

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>

<body>
    <div class="header">
    </div>
    <section id="container">
        <ul>
            <li class="first">uno</li>
            <li class="second">dos</li>
            <li class="third">tre</li>
        </ul>
        <ol>
            <li class="first">uno</li>
            <li class="second">dos</li>
            <li class="third">tres</li>
        </ol>
    </section>
    <div class="footer">
    </div>
    <script src="ex03.js"></script>
</body>
</html>
```

3.a – Seleccionar la seccion con el Id container sin utilizar querySelector. Y cámbiela dinámicamente despues de 3 segundos.
3.b – Seleccionar todos los list items con la clase "second" . Y cámbie los elementos dinámicamente despues de 3 segundos.
3c – haciendo un loop de todos los elementos <li> dentro de haga que tengan un color de background "green"

3d - Remover el div con la clase footer

4. Haga una pagina html que muestre una alarma configurable . Debe mostrar un reloj y permitir configurar una alarma que muestre un mensaje de alerta cuando se lance.