A continuación, aprenderemos a utilizar métodos nativos de Javascript. Hay muchísimos. Nos vamos a detener en dos porque tienen utilidades prácticas muy evidentes. Son "setTimeout" y "setInterval". El primero ejecuta algo, una función, una serie de instrucciones, luego de transcurrido un tiempo determinado. Determinado por nosotros. Eso hace "setTimeout". "SetInterval", por el contrario, ejecuta también una instrucción, una serie de instrucción, de instrucciones, lo que sea, una vez cada 'n' segundos o milisegundos. Es decir, es una estructura que repite nuestro código según el tiempo o el lapso que nosotros indicamos. Vamos a ver entonces, "setTimeout" y "setInterval". Métodos nativos de javascript en el código. Vamos a comenzar con "setTimeout". Esta es la sintaxis.

Recibe un callback y con callback, esa función va a ejecutar algún proceso, alguna algún funcionalidad. Y luego de la coma, aquí, vamos a indicar un tiempo en milisegundos. 2000 milisegundos son, bueno, dos segundos. Y aquí, podríamos indicar, simplemente, vamos a imprimir algo, vamos a imprimir "mundo". Luego del setTimeout podríamos imprimir "hola". Notara que al ejecutar javascript, esta línea, no se ejecutará durante cuarenta. Seguirá ejecutando el código siguiente. Por lo tanto, esta línea no va a imprimir antes que esta. Porque esta que está marcada aquí, el "console.log ("mundo")" debe ejecutarse una vez que hayan transcurridos, que hayan transcurrido, los dos segundos, los 2000 milisegundos. Por lo tanto, si corremos esta porción de código, de la línea 3 a la línea 6, el resultado será "hola", y luego de dos segundos, "mundo". Vamos a verlo aquí en la consola. "hola" "mundo"

¿ok? Muy bien, como habrán notado ustedes, si este console.log.o estuviese aquí arriba, el resultado sería exactamente el mismo. Ya que esta porción está ejecutando código de manera asincrónica. Es decir, lo que está aquí adentro en el callback se va a ejecutar luego de transcurrido este tiempo. Todo lo que está luego de este setTimeout seguirá contando de manera normal sin aguardar absolutamente nada. Ok. Ahora, podría ser que nosotros deseamos interrumpir la ejecución de este código. Por ejemplo, precisamente no hubiéramos de que esto se imprimiera. Todavía no manejamos el **dom**, lo vamos a hacer en un par de clases. Por lo tanto, no vamos a poner aquí un botón para precisarlo, pero vamos a mostrar como podríamos evitar que este efectivamente corra, se ejecute. Tenemos un método que se llama "clearInterval" y recibe el nombre del intervalo. Es decir, si nosotros comenzó esto aquí no vamos a obtener ningún resultado porque deberíamos asignar esta función a alguna variable que luego podamos pasar aquí adentro como parámetro. Entonces, podríamos hacer, por ejemplo, esta constante que se llama 'timer'. Bien, noten que el ejemplo este,

aquí no va a haber ningún cambio. Va a seguir ejecutándose este código, pero si aquí nosotros ahora, hacemos un clearInterval y decimos que el intervalo que queremos limpiar o eliminar es este, se va a detener esta función antes de que lleguen los 2 segundos. Es decir, esto nunca va a correr. Ok. Perdón, aquí va el nombre 'timer'. Y ahora sabrán con a ver en pantalla que esta línea se imprime, pero luego limpiamos el intervalo y esto nunca llega a imprimirse porque interrumpimos el proceso. Mirar. "hola", contamos dos segundos, y tres, y cuatro, y cinco, y se nos va el tren. Nunca se va a imprimir esto porque hemos interrumpido la ejecución. Ok. Ahora setInterval hace algo totalmente distinto. SetInterval lo que hace es, es, bueno, está aquí determinado ¿verdad? Aquí lo tenemos lo tenemos descrito, antes de pasar de empezar la clase lo he descrito. Lo que hace es ejecutar una función o un fragmento de código de forma repetitiva. Es decir, también tendríamos un reloj pero un reloj no indicará, ok, ejecuta todo lo que está aquí adentro pasado este tiempo. SetInterval dice: "ejecuta todo lo que está aquí adentro en este lapso". Es decir, de manera repetitiva cada 'n' segundos, bien, o milisegundos en este caso, ¿no?. Podríamos hacerlo así, setInterval

La sintaxis es idéntica al setTimeout y aquí podríamos decir