**Отложенные действия**

Для того, чтобы выполнить какое-то действие спустя некоторое время, используются специальные функции **setTimeout()** и **setInterval().**

**setTimeout()**

Данная функция выполняется один раз, через определенное количество миллисекунд.

setTimeout(callback, time)

**callback**- функция обратного вызова.  
**time** - время задержки в миллисекундах.  
Пример:

setTimeout(() => {

console.log('1 second passed')

}, 1000)

Мы вывели в консоль надпись '1 second passed', через секунду после запуска функции, один раз.

Бывают случаи, когда мы запустили setTimeout, но нам он нам стал не нужен. Тогда мы можем отменить его работу с помощью **timeoutId** (можно переименовать) и функции **clearTimeout()**.  
Пример:

const timeoutId = setTimeout(() => {

console.log('2 second passed')

}, 2000)

setTimeout(() => {

console.log('1 second passed')

clearTimeout(timeoutId)

}, 1000)

В консоли мы увидим только '1 second passed', так как отменили работу первой setTimeout(), которая должна была вывести свой текст через 2 секунды.

**setInterval()**

Данная функция будет выводить результаты работы callback-функции через заданный интервал времени, без остановки.

setInterval(callback, time)

**callback** - функция обратного вызова.  
**time** - время задержки в миллисекундах.  
Пример:

setInterval(() => {

console.log('1 second passed')

}, 1000)

Каждую секунду в консоли будет выводиться строка '1 second passed'.

Для отмены работы setInterval используется функция **clearInterval(intervalId)**, где intervalId - это идентификатор отменяемого интервала.

Пример:

setInterval(() => {

console.log('2 second passed')

clearInterval(intervalId)

}, 2000)

const intervalId = setInterval(() => {

console.log('1 second passed')

}, 1000)

В консоли будет одна строка '1 second passed', так как мы отменили работу данного интервала. Но каждые 2 секунды будет выводиться строка '2 second passed'.

**Внимание!**

Стоит понимать, что функция console.log() во всех примерах выше является исключительно примером для наглядности и вам не нужно её постоянно использовать для описанных выше функций.

Вы можете записывать в callback абсолютно любую нужную вам функцию!

Код для урока

const throttledFunc = (func, interval) => {

let date = +new Date()

return () => {

if (date < +new Date() - interval) {

date = +new Date()

func()

}

}

}

const newFunc = throttledFunc(() => {display.log(new Date())}, 3000)