Edgemony

Coding Bootcamp

Gli intervalli temporali

setTimeout e setInterval, una parvenza di asincronicità

*Time **and** space **are not conditions of existence, time and space is a model of thinking.***

- Albert Einstein

setTimeout – fermi tutti…

Il metodo **setTimeout** accetta due parametri:

1. la **funzione** che verrà lanciata;
2. Il **tempo** (espresso in millisecondi: 1000m === 1s) che impiegherà per lanciarla (la funzione!) e infine auto-concludersi.

```
setTimeout(function(){ console.log('Ciao… a scoppio ritardato!') }, 3000);
setTimeout(() => console.log('Ciao… a scoppio ritardato!'), 3000);

// STESSO IDENTICO RISULTATO!
```

Il metodo ritorna anche un numero, questo rappresenta il proprio ID (codice identificativo), il passaporto del timer all'interno del nostro programma). Indispensabile per identificare questo specifico timer e, in caso, interromperne l'esecuzione.

Da notare: la callback!

clearTimeout – dietro ogni grande metodo…

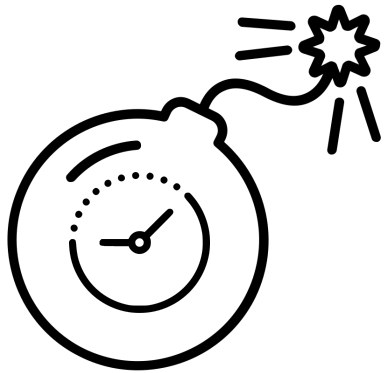
Il metodo **setTimeout** presenta una controparte che lo '**annulla**'.

Dato il seguente: `setTimeout(() => console.log('Ciao… a scoppio ritardato!'), 3000);`

Possiamo interromperne il timeout, innescando:

```
clearInterval(ID);
clearInterval(varName);

// STESSO IDENTICO RISULTATO!
```



Attenzione: lanciare la bomba va disinnescata prima che esploda!

setInterval – ogni giorno, alle stessa ora, il sole sorge

Il metodo **setInterval** accetta anch'esso due parametri:

1. la **funzione** che verrà lanciata
2. Il **tempo** (espresso in millisecondi) che impiegherà per lanciarla, ogni tot.

```
setInterval(function(){  console.log('Ciao…  ogni  3  secondi!')  },  3000);
setInterval(() => console.log('Ciao… ogni 3 secondi!'), 3000);

// STESSO IDENTICO RISULTATO!
```

Se dunque setTimeout esegue solamente una funzione al termine di quel lasso temporale e poi si auto-distrugge, setInterval non si arresterà, anzi, continuerà anche all'infinito!

clearInterval – a tutto c'è una fine.

Anche il metodo **setInterval** può essere interrotto e, in effetti, in questo caso ha anche più 'senso' logico del **setItemeout**. Il comportamente è del tutto analogo.

Dato il seguente: `setInterval(() => console.log('Ciao… ogni 3 secondi!'), 3000);`

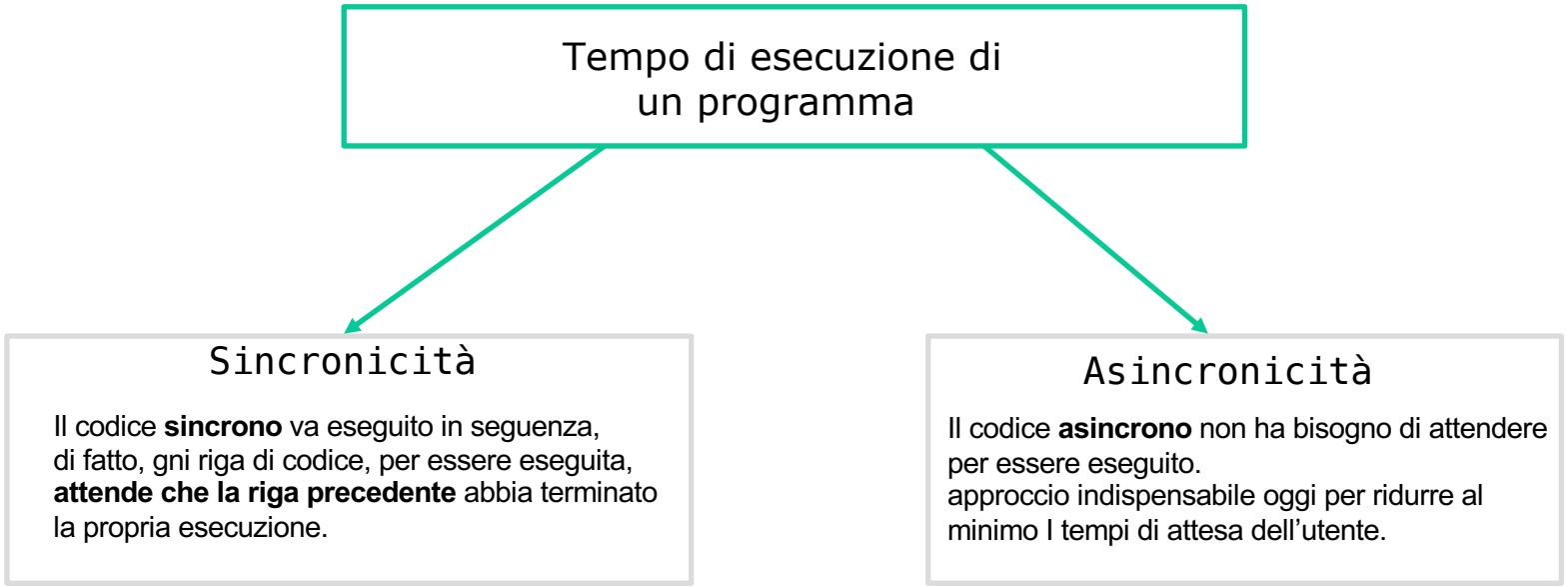
Possiamo interromperne il timeout, innescando:

```
clearTimeout(ID);
 clearTimeout(varName);

// STESSO IDENTICO RISULTATO!
```

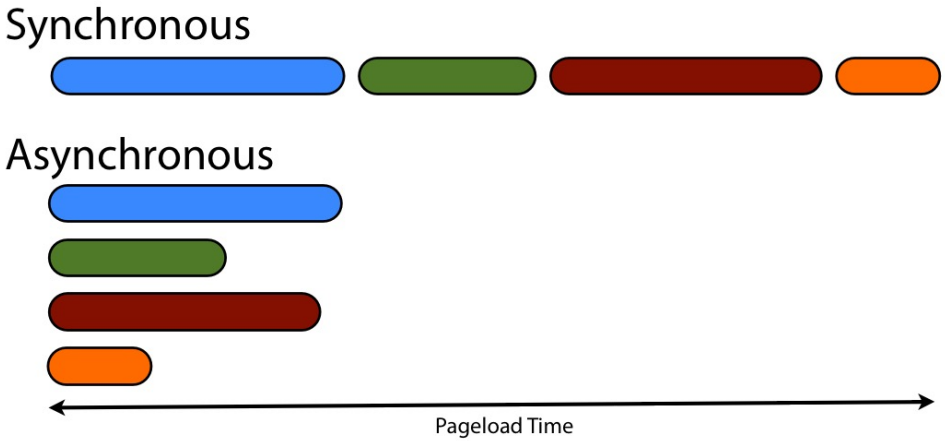
Attenzione: qui possiamo lasciare la bomba esploda e poi interromperla dopo

Una introduzione veloce veloce alla programmazione asincrona



Gli intervalli temporali

setTimeout e setInterval, una parvenza di asincronicità



Gli intervalli temporali

setTimeout e setInterval, una parvenza di asincronicità