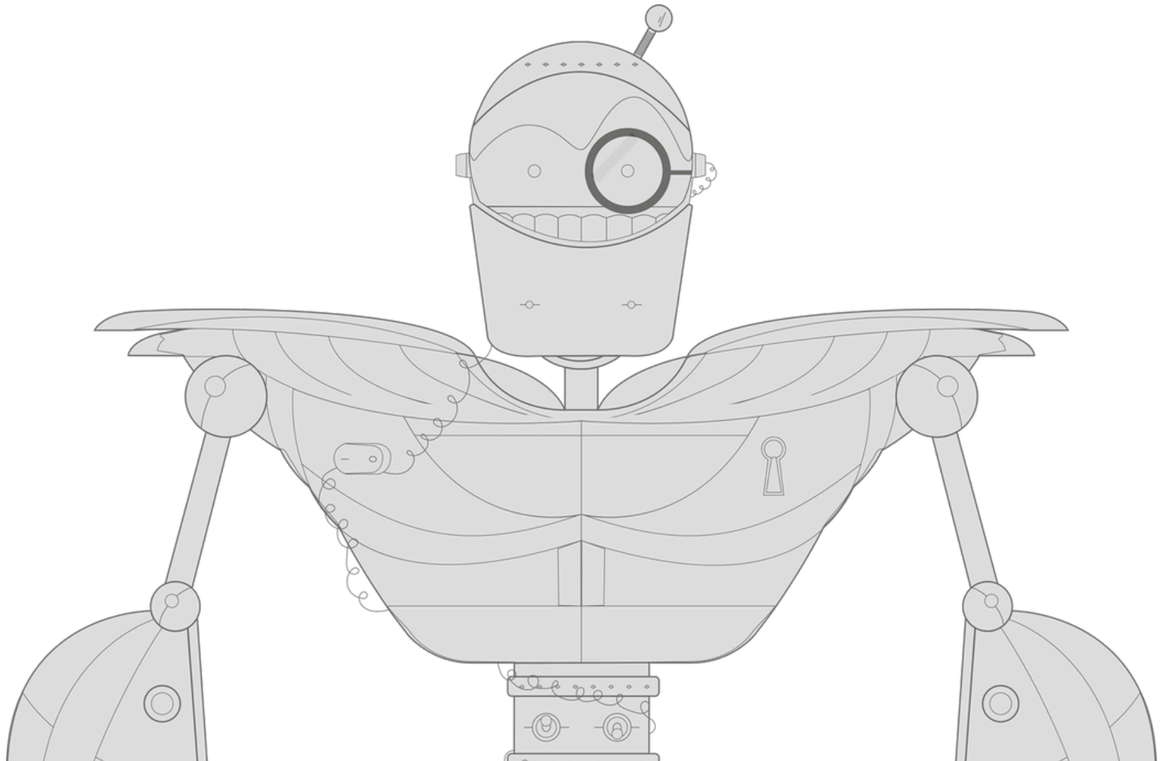
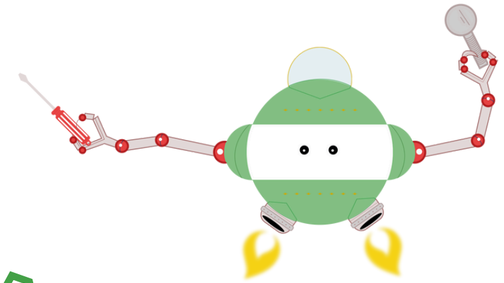
Labforweb
nerd academy

setTimeout() & setInterval()

A cura di Gaia Moschetto





IN 1/2 'ORA

Cosa sono?

↪ setTimeout()

È un *metodo* JavaScript che permette di eseguire (*una sola volta*) una funzione dopo un intervallo di tempo prescelto. Occorre passargli ***2 parametri***:

- il primo è la funzione che vogliamo venga eseguita;
- il secondo è l'intervallo di tempo dopo il quale vogliamo che la funzione venga eseguita.

N.B.: L'intervallo di tempo è espresso in millisecondi!

```
var myTimer = setTimeout(
  function() {console.log("test")}, 5000
);
```

*Il metodo setTimeout() crea un timer che eseguirà la funzione specificata come primo parametro **dopo** 5000 millisecondi. Quello che otterremo è la visualizzazione della stringa "test" sulla console dopo 5 secondi.*

↪ setTimeout()

Se intendiamo fermare l'esecuzione della funzione *prima* che venga eseguita, dobbiamo utilizzare il metodo **clearTimeout()** come mostrato di seguito:

```
var myTimer = setTimeout(function() {console.log("test")}, 5000);

clearTimeout(myTimer);  //non accade niente, perchè ho fermato l'esecuzione
```

⚠ ...attenzione! ⚠

Spesso viene commesso questo errore:

```
function saluta() {
  alert('Ciao');
}

setTimeout(saluta(), 1000);
```

Qual è l'errore?

In questo modo la funzione passata come parametro verrà eseguita *immediatamente*, e non dopo 1 secondo. Questo perchè all'interno del setTimeout() non si deve ***invocare*** la function, ma occorre ***riferirsi*** ad essa!

Come risolvere?

```
function saluta() {
  alert('Ciao');
}

setTimeout(saluta, 1000); //no parentesi tonde quando si richiama la function saluta
```

Cosa sono?

↪ setInterval()

Anch'esso è un *metodo* JavaScript che viene usato per eseguire una funzione *ad intervalli regolari*. La sintassi è uguale a quella di setTimeout(), quindi anche qui è previsto l'utilizzo di ***2 parametri***:

- il primo è la funzione che vogliamo venga eseguita;
- il secondo è l'intervallo di tempo che vogliamo far trascorrere tra una chiamata di funzione e l'altra.

```
var myTimer = setInterval(
  function() {console.log("test")}, 5000
);
```

Il metodo setInterval() crea un timer che eseguirà la funzione specificata come primo parametro ***ogni*** *5000 millisecondi. Quello che otterremo è la visualizzazione della stringa "test" sulla console ogni 5 secondi.*

↪ setInterval()

Se vogliamo evitare altre chiamate di funzione, dobbiamo eseguire il metodo **clearInterval()** come mostrato di seguito:

```
var myTimer = setInterval(function() {console.log("test")}, 5000);

clearInterval(myTimer)
```

⚠ ...attenzione! ⚠

Spesso viene commesso questo errore:

```
function saluta() {
  alert('Ciao');
}

setInterval(saluta(), 1000);
```

Qual è l'errore?

In questo modo la funzione passata come parametro verrà eseguita *immediatamente,* e non dopo 1 secondo. Questo perchè all'interno del setInterval() non si deve ***invocare*** la function, ma occorre ***riferirsi*** ad essa!

Come risolvere?

```
function saluta() {
  alert('Ciao');
}

setInterval(saluta, 1000);
```

Ricapitolando:

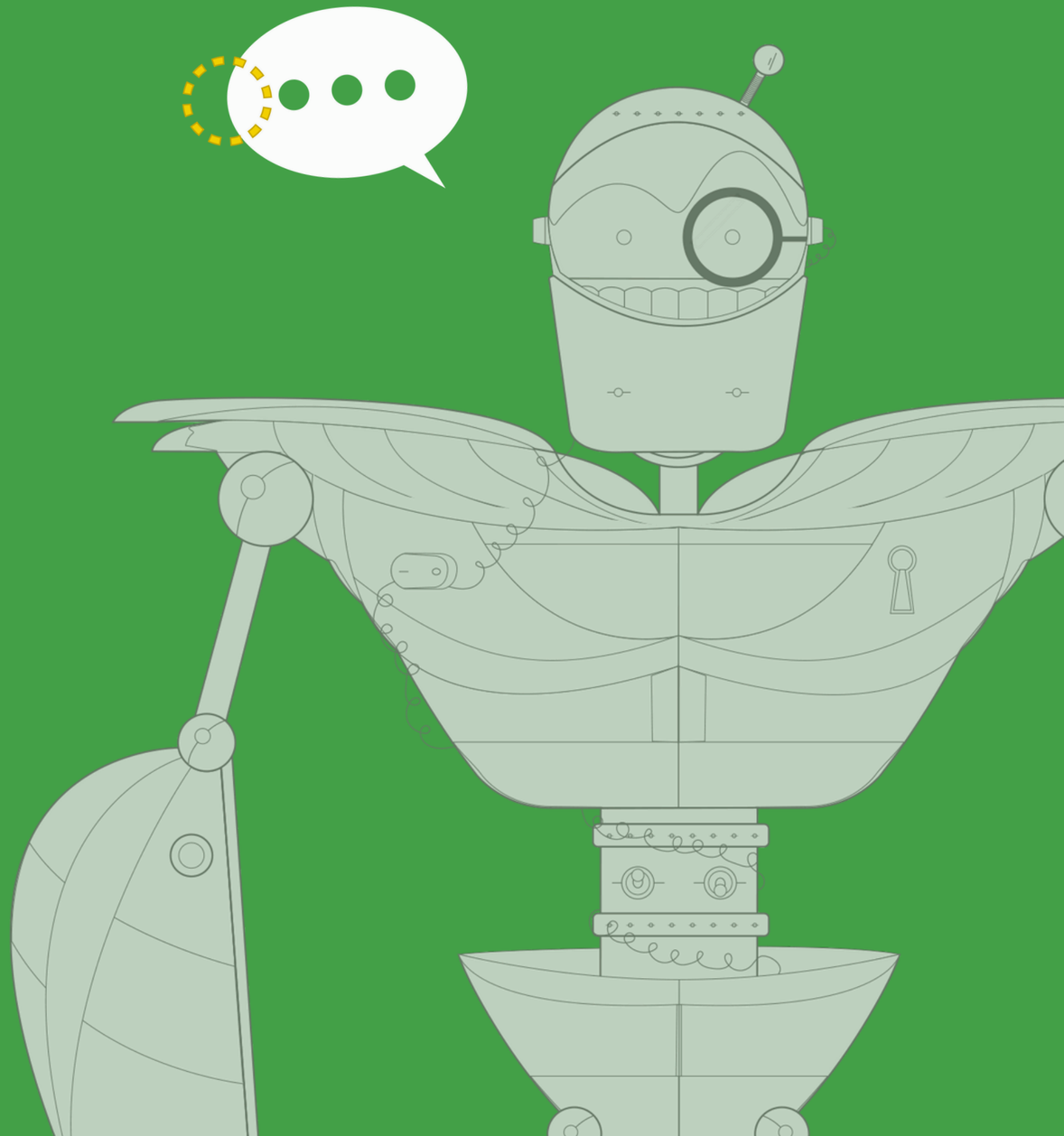
- I metodi *setInterval(**func**, **ritardo**)* e *setTimeout(**func**, **ritardo**)* consentono di avviare la func *a intervalli regolari* nel primo caso, o *una sola volta* dopo un certo ritardo nel secondo caso.

- Per *disattivare* l'esecuzione, dovremo chiamare *clearInterval() o clearTimeout()* con il valore restituito rispettivamente da setInterval() e setTimeout().

  Da notare che i metodi di pianificazione non garantiscono un ritardo preciso.
  Per esempio, il timer nel browser può rallentare per molte ragioni:
  - La CPU è sovraccarica.
  - La scheda del browser è sullo sfondo.
  - Il portatile ha la batteria scarica.

Labforweb
nerd academy

Domande...



Grazie

www.labforweb.it