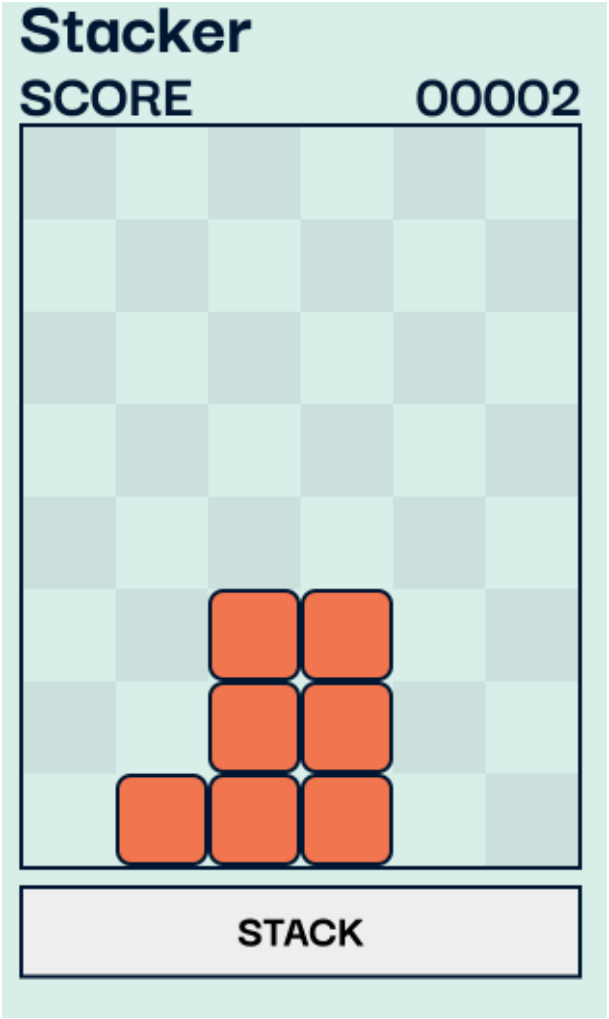

boolean

Stacker

"Go stack yourself!"

Cosa impariamo:

- qualche concetto di game design
- nuovi metodi degli array
- timing functions



Game Loop

update and draw

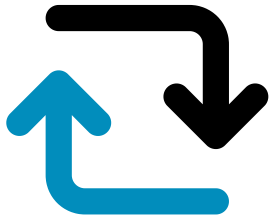
Game Loop

Ciclo di gioco

E' un **ciclo continuo** che esegue ripetutamente una serie di operazioni per aggiornare lo **stato del gioco** e **renderizzare** i cambiamenti su schermo.

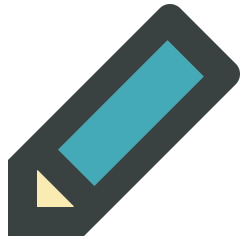
Input

Cattura gli input del giocatore per comandare il gioco (es. pressione di un tasto)



Update

Calcola i cambiamenti nello stato del gioco in base agli input del giocatore e alle regole del gioco (es. movimento personaggio, collisioni etc.)



Draw

Disegna gli elementi del gioco sullo schermo in base allo stato aggiornato.

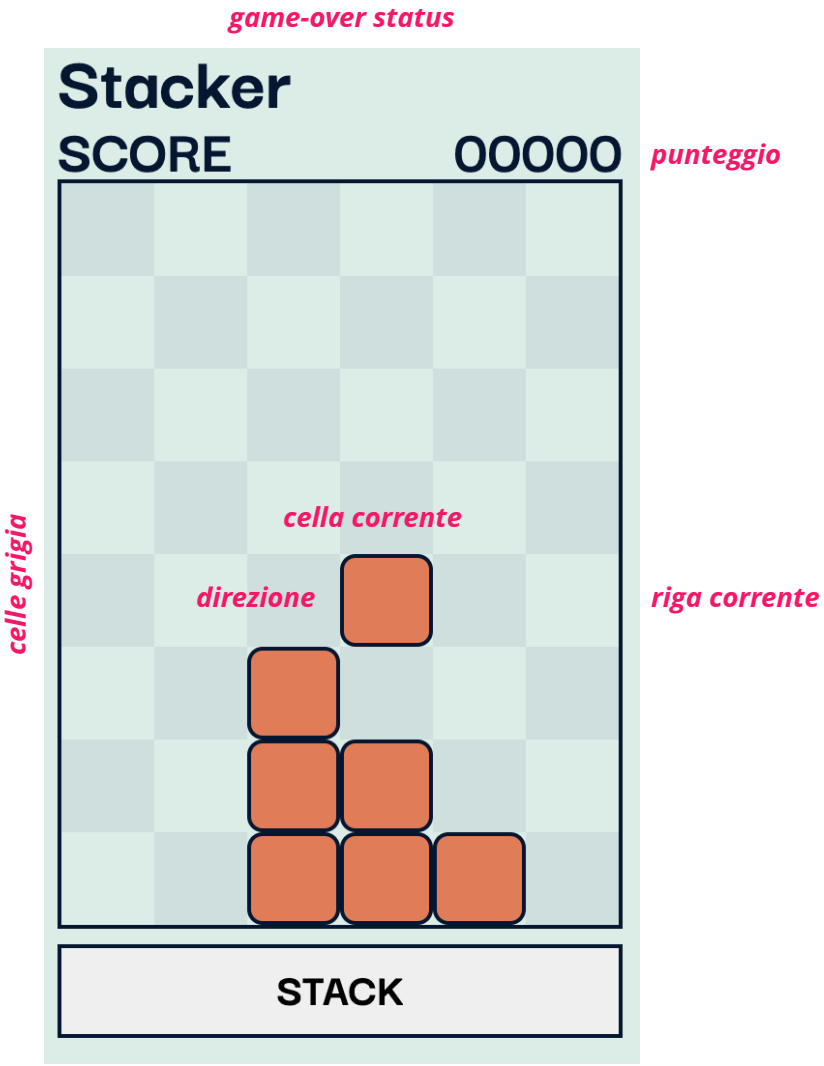
E' l'aspetto visuale del gioco.

Game State

Stato del gioco

E' la rappresentazione dei dati che definiscono lo stato di un videogioco in un determinato momento.

Questi dati includono informazioni come la posizione dei personaggi, lo stato degli oggetti, il punteggio del giocatore, le condizioni di vittoria o perdita e molti altri dettagli relativi al progresso del gioco.



Tick

Un ciclo del loop

Un tick rappresenta un ciclo (o un'unità di tempo) all'interno del game loop, durante il quale vengono eseguite una serie di operazioni quali update dello stato e render.

Queste possono quindi includere:

- aggiornamento della posizione dei personaggi e degli oggetti
- elaborazione degli input del giocatore
- verifica delle condizioni di vittoria o perdita
- aggiornamento delle animazioni e della grafica

Ciclo forEach

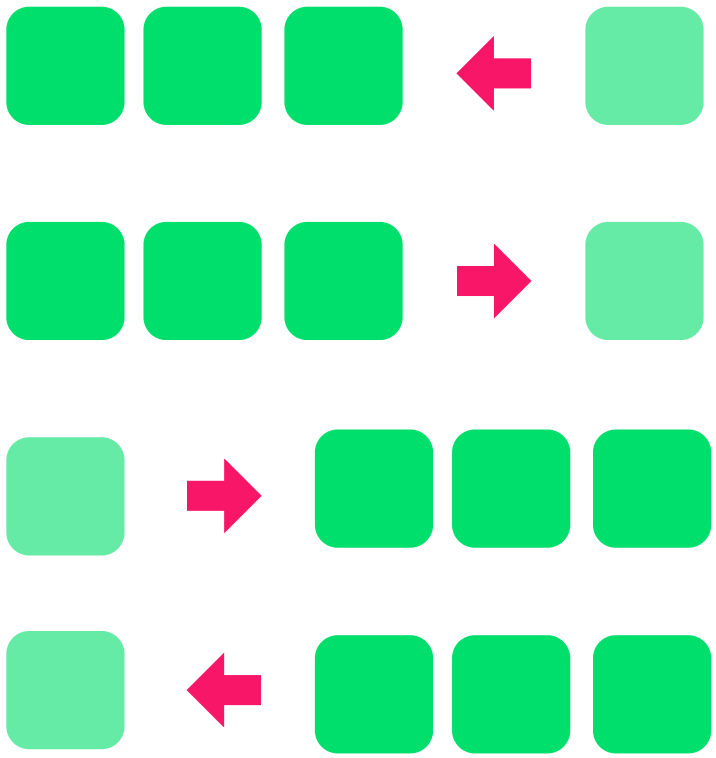
Un nuovo vecchio ciclo

forEach ci aiuta a ciclare sugli array in maniera più concisa

- **keyword**
  forEach
- **function**
  All'interno delle parentesi tonde inseriamo una funzione che riceverà come argomenti:
  element - l'elemento dell'array sul quale stiamo girando
  index - l'indice di quell'elemento
  array - l'array stesso
- **codice:**
  che verrà eseguito ad ogni giro

```
1  const students = ['Paolo', 'Giulia', 'Marco'];
2
3  students.forEach(function(element, index, array) {
4   console.log(element, index);
5  });
6
7
8  // Paolo 0
9  // Giulia 1
10 // Marco 2
```

Altri metodi per manipolare un array



.push()

aggiunge elementi alla fine di un array

.pop()

rimuove un elemento dalla coda di un array

.unshift()

aggiunge elementi all'inizio di un array

.shift()

rimuove elementi dalla testa di un array

setInterval

Facciamo qualcosa **ogni tot**

2 **setInterval** invoca una funzione **all'infinito, ogni tot di tempo**.

setInterval(nomeFunzione, tempoPerOgniRipetizione);

1. invoca una funzione
2. che si verrà eseguita ogni "n" millisecondi

```
1   const clock = setInterval(myFunction, 3000);
2
3   function myFunction() {
4      alert('Hello');
5   }
```

setInterval

Facciamo qualcosa **ogni tot**

Come fermo un setInterval?

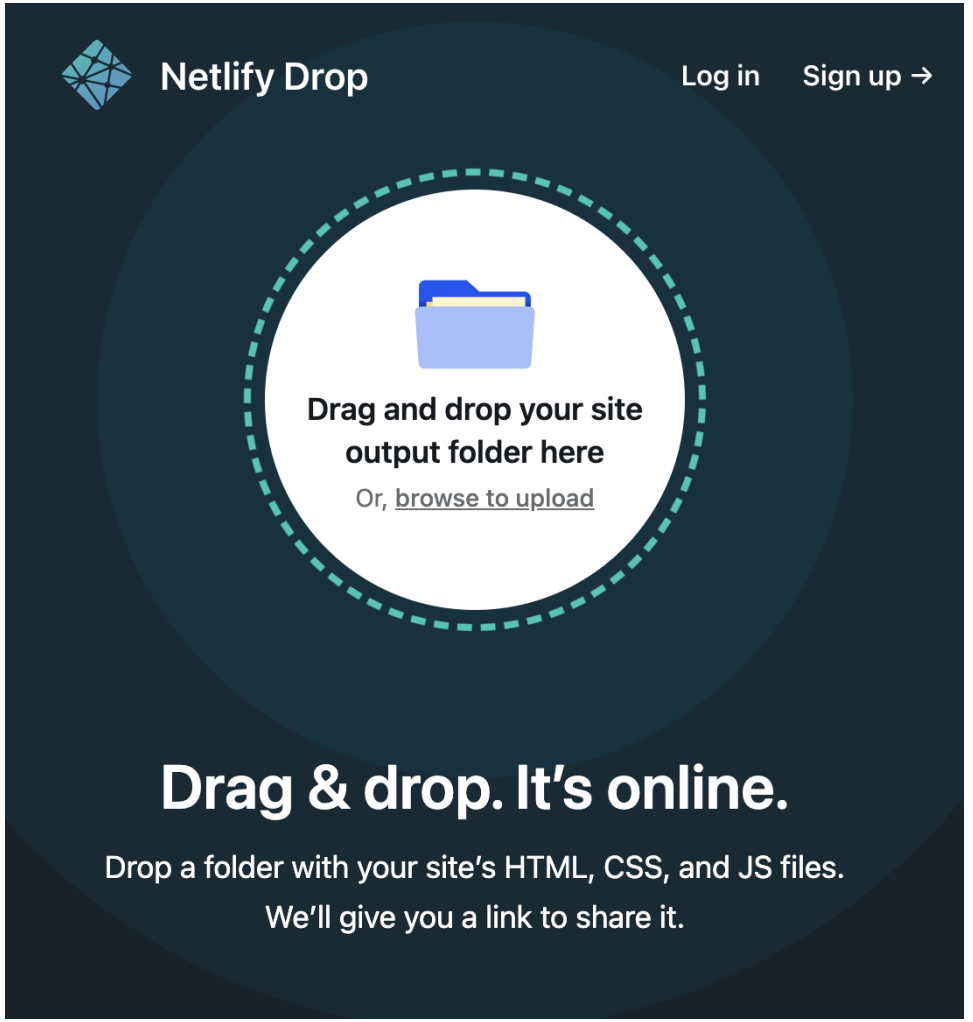
1. Salvo il risultato del **setInterval** in una **variabile**
2. La passo alla funzione:
   clearInterval(nomeVariabileSetInterval)

```
1  const clock = setInterval(myFunction, 3000);
2  //console.log(clock) <- cosa ci restituirà?
3
4  function myFunction() {
5   console.log('Hello');
6  }
7
8  clearInterval(clock);
```

Deploy

Mettiamo il nostro codice su un server

https://app.netlify.com/drop



CHALLENGE

Game bonus!

The higher, the faster

Aggiungi dei bonus al gameplay!

Ecco alcune idee:

Moltiplicatori

Attiva un moltiplicatore di punti e aumenta la velocità di gioco quando lo stack raggiunge una certa altezza.

Sblocca scenari extra

Puoi anche decidere di alterare colori/sfondo della scena man mano che ti avvicini alla cima o perdi blocchi.