# DOCUMENTAZIONE TETRIS

```
1  const cvs = document.getElementById("tetris");
2  const ctx = cvs.getContext("2d");
3  const EPunteggio = document.getElementById("punteggio");
4
5  const RIG= 20;
6  const COL= 10;
7  const SQ = 20;
8  const SV = "white";
9  const Pezzi = [
10     [Z,"red"],
11     [S,"green"],
12     [T,"yellow"],
13     [O,"blue"],
14     [L,"purple"],
15     [I,"cyan"],
16     [J,"orange"]
17 ];
```

1. cvs = costante a cui attribuiamo il canvas,
2. cvs.getContext("2d") -> questa è la funzione che viene utilizzata per accedere alle funzioni di disegno 2D dei tag canvas.
3. RIG, COL, SQ, SV, Pezzi -> definiscono il numero di righe e colonne del tetris, la dimensione dei quadrati e il colore di sfondo mentre Pezzi definisce il colore delle forme.

```
20 var img = document.createElement("img");
21 var song = true;
22 let punteggio = 0;
23 var sottofondo = document.createElement('audio');
24 var perso = document.createElement('audio');
```

4. Dichiarazione di variabili varie

```
27 $(".game").hide();
28 setInterval(start, 2000);
29 function start(){
30     $(".game").show();
31     $("div:first").hide();
32 }
```

5. Tutti gli elementi della pagina HTML hanno la classe ".game". Utilizzando il metodo ".hide", abbiamo la possibilità di nascondere tutto il contenuto della pagina, tranne la barra di caricamento. Dopo due secondi, il gioco viene visualizzato, il tempo necessario per completare la barra di caricamento.

```
35 function disegnaQuad(x,y,color){
36     ctx.fillStyle = color;
37     ctx.fillRect(x*SQ,y*SQ,SQ,SQ);
38
39     ctx.strokeStyle = "BLACK";
40     ctx.strokeRect(x*SQ,y*SQ,SQ,SQ);
41 }
```

6. La funzione disegnaQuad disegna una forma del colore specificato nelle coordinate (x*SQ, y*SQ) con dimensioni SQ x SQ. Inoltre, disegna un bordo nero intorno al quadrato utilizzando il metodo strokeRect.

```
44  let tabella = [];
45  for( r = 0; r <RIG; r++){
46      tabella[r] = [];
47      for(c = 0; c < COL; c++){
48          tabella[r][c] = SV;
49      }
50  }
51
52  function disegnaTab(){
53      for( r = 0; r <RIG; r++){
54          for(c = 0; c < COL; c++){
55              disegnaQuad(c,r,tabella[r][c]);
56          }
57      }
58  }
59
60  disegnaTab();
```

7. La funzione disegnaTab utilizza il metodo disegnaQuad, definito in precedenza, per disegnare il quadrato nella tabella.
   Scorre attraverso la tabella utilizzando due cicli for annidati, uno per le righe e uno per le colonne.
   Ad ogni iterazione, chiama la funzione disegnaQuad per disegnare l'oggetto nella posizione specificata dalle variabili r e c con il colore specificato nell'array tabella[r][c].

```
65  function PezziRandom(){
66      let r = Math.floor(Math.random() * Pezzi.length) // 0 -> 6
67      return new Pezzo( Pezzi[r][0],Pezzi[r][1]);
68  }
69
70  let p = PezziRandom();
```

8. La funzione "PezziRandom()" genera un numero casuale che viene utilizzato per selezionare casualmente un elemento all'interno dell'array.
   Questo elemento viene quindi passato come argomento alla funzione "Pezzo(tetramino, color)" per creare un nuovo oggetto "Pezzo".
   Il "return new Pezzo" restituisce un nuovo oggetto "Pezzo" random, utilizzando un indice random per selezionare il tetramino e il colore corrispondente dalla matrice "Pezzi".

```
75  function Pezzo(tetramino,color){
76      this.tetramino = tetramino;
77      this.color = color;
78
79      this.tetraminoN = 0;
80      this.TetraminoAttivo = this.tetramino[this.tetraminoN];
81
82      this.x = 3;
83      this.y = -2;
84
85  }
```

9. La funzione 'Pezzo' viene utilizzata per creare una nuova forma tetramino.
   Nello specifico "this.tetraminoN = 0;" si riferisce all'indice del tetramino attualmente in uso, mentre il "this.TetraminoAttivo" si riferisce all'effettiva forma in uso.

```
88  Pezzo.prototype.riempi = function(color){
89      for( r = 0; r < this.TetraminoAttivo.length; r++){
90          for(c = 0; c < this.TetraminoAttivo.length; c++){
91              if( this.TetraminoAttivo[r][c]){
92                  disegnaQuad(this.x + c,this.y + r, color);
93              }
94          }
95      }
96  }
```

10. Aggiungiamo un nuovo metodo "riempi" all'oggetto "Pezzo" utilizzando la proprietà "prototype" che viene utilizzato per disegnare il tetramino sul tabellone di gioco controllando che lo stesso tabellone non sia già occupato da altri elementi.

```
99  Pezzo.prototype.disegna = function(){
100     this.riempi(this.color);
101 }
```

11. Aggiungiamo un nuovo metodo "disegna" all'oggetto "Pezzo" utilizzando la proprietà "prototype" che usa il metodo “riempi” per disegnare il quadrato con il colore corrente.

```
104     Pezzo.prototype.cancella = function(){
105         this.riempi(SV);
106     }
```

12. Aggiungiamo un nuovo metodo "cancella" all'oggetto "Pezzo" utilizzando la proprietà "prototype" che invece riempie i quadrati dell'area di gioco con il colore di SV, cancellando così il tetramino precedentemente disegnato usando il metodo “.riempi”

```
109 Pezzo.prototype.muoviGiu = function(){
110     if(!this.collisione(0,1,this.TetraminoAttivo)){
111         this.cancella();
112         this.y++;
113         this.disegna();
114     }else{
115         this.blocca();
116         p = PezziRandom();
117     }
118
119 }
120 //
121
122 Pezzo.prototype.muoviDestra = function(){
123     if(!this.collisione(1,0,this.TetraminoAttivo)){
124         this.cancella();
125         this.x++;
126         this.disegna();
127     }
128 }
129 //
130
131 Pezzo.prototype.muoviSinistra = function(){
132     if(!this.collisione(-1,0,this.TetraminoAttivo)){
133         this.cancella();
134         this.x--;
135         this.disegna();
136     }
137 }
138 //
139
140 Pezzo.prototype.tuttoGiu = function(){
141     while(!this.collisione(0,1,this.TetraminoAttivo)){
142         this.cancella();
143         this.y++;
144         this.disegna();
145     }
146 }
```

13. Aggiungiamo un nuovo metodo "muoviGiu" all'oggetto "Pezzo" utilizzando la proprietà "prototype" che ha lo scopo di muovere verso il basso il tetramino attivo se non vi è una collissione, altrimenti ferma il tetramino attivo nel punto di collissione e ne crea uno nuovo.

14. Aggiungiamo un nuovo metodo "muoviDestra" all'oggetto "Pezzo" utilizzando la proprietà "prototype" che ha lo scopo di muovere verso destra il tetramino attivo se non vi è una collissione.

15. Aggiungiamo un nuovo metodo "muoviSinistra " all'oggetto "Pezzo" utilizzando la proprietà "prototype" che ha lo scopo di muovere verso sinistra il tetramino attivo se non vi è una collissione.

16. Aggiungiamo un nuovo metodo "tuttoGiu " all'oggetto "Pezzo" utilizzando la proprietà "prototype" che ha lo scopo di spostare verso il basso, finché non incontra una collisione, il tetramino attivo.

se non ci sono collisioni, spostano l'oggetto e ridisegnano il tetramino con la nuova posizione utilizzando le funzioni cancella e disegna.

Se invece c'è una collisione, la funzione blocca viene chiamata per bloccare l'oggetto nella posizione corrente e viene generato un nuovo oggetto Pezzo casuale utilizzando la funzione PezziRandom.

```
149  Pezzo.prototype.ruota = function(){
150      let prossimoPezzo = this.tetramino[(this.tetraminoN + 1)%this.tetramino.length];
151      let sposta = 0;
152
153      if(this.collisione(0,0,prossimoPezzo)){
154          if(this.x > COL/2) sposta = -1;
155          else sposta = 1;
156      }
157
158      if(!this.collisione(sposta,0,prossimoPezzo)){
159          this.cancella();
160          this.x += sposta;
161          this.tetraminoN = (this.tetraminoN + 1)%this.tetramino.length;
162          this.TetraminoAttivo = this.tetramino[this.tetraminoN];
163          this.disegna();
164      }
165  }
```

17. Aggiungiamo un nuovo metodo "ruota" all'oggetto "Pezzo" utilizzando la proprietà "prototype" che ha la funzione di ruotare il pezzo corrente in senso orario. Considerando la possibilità di una collisione il metodo cerca di spostare il pezzo di una posizione a destra o a sinistra in modo da evitare la stessa.
    Per fare questo, viene controllata la posizione del pezzo e viene deciso in quale direzione spostare il pezzo per evitare la collisione. Se la posizione del pezzo è a destra della metà della larghezza del gioco, allora il pezzo viene spostato a sinistra, altrimenti viene spostato a destra.
    Se non c'è collisione, il metodo "ruota" cancella il pezzo aggiornandone la posizione e il valore "tetraminoN" per il nuovo pezzo attivo, e infine disegna il pezzo nella sua nuova posizione.

```
218  Pezzo.prototype.collisione = function(x,y,pezzo){
219      for( r = 0; r < pezzo.length; r++){
220          for(c = 0; c < pezzo.length; c++){
221              if(!pezzo[r][c]){
222                  continue;
223              }
224              let newX = this.x + c + x;
225              let newY = this.y + r + y;
226
227              if(newX < 0 || newX >= COL || newY >= RIG){
228                  return true;
229              }
230
231              if(newY < 0){
232                  continue;
233              }
234
235              if( tabella[newY][newX] != SV){
236                  return true;
237              }
238          }
239      }
240      return false;
241  }
```

18. Aggiungiamo un nuovo metodo "collisione" all'oggetto "Pezzo" utilizzando la proprietà "prototype" prende tre argomenti: "x" e "y", che rappresentano la posizione dell'oggetto, e "pezzo", che rappresenta il tipo di pezzo da controllare per le collisioni.
    I due cicli hanno il compito di calcolare la nuova posizione del pezzo aggiungendo "c" e "r" (gli indici del ciclo) alle coordinate correnti dell'oggetto, insieme ai valori di "x" e "y".
    La funzione controlla poi se le nuove coordinate appena calcolate sono al di fuori dei limiti del canvas, restituendo true se lo sono. Infine, la funzione controlla se l'elemento alle nuove coordinate della matrice "tabella" è uguale a "SV". Se non lo è, la funzione restituisce true, indicando che si è verificata una collisione, al contrario, la funzione restituisce false, indicando che non ci sono collisioni.

```
170  Pezzo.prototype.blocca = function(){
171      for(r = 0; r < this.TetraminoAttivo.length; r++){
172          for(c = 0; c < this.TetraminoAttivo.length; c++){
173
174              if(!this.TetraminoAttivo[r][c]){
175                  continue;
176              }
177
178              if(this.y + r < 0){
179                  $("canvas").remove();
180                  img.src = "images/tetris/gameover.png";
181                  var src = document.getElementById("gameover");
182                  src.appendChild(img);
183                  perso.setAttribute('src', 'sounds/gameover.mp3');
184                  perso.play();
185                  sottofondo.pause();
186                  $("img:first").css("margin-top","180px");
187                  EPunteggio.style.marginTop = -170 + "px";
188                  gameOver = true;
189                  break;
190              }
191              else tabella[this.y+r][this.x+c] = this.color;
192          }
193      }
194      for(r = 0; r < RIG; r++){
195          let RigaPiena = true;
196          for(c = 0; c < COL; c++){
197              RigaPiena = RigaPiena && (tabella[r][c] != SV);
198          }
199          if(RigaPiena){
200              for(y = r; y > 1; y--){
201                  for( c = 0; c < COL; c++){
202                      tabella[y][c] = tabella[y-1][c];
203                  }
204              }
205              for(c = 0; c < COL; c++){
206                  tabella[0][c] = SV;
207              }
208              punteggio += 10;
209          }
210      }
211      disegnaTab();
212      EPunteggio.innerHTML = punteggio;
213  }
```

19. Aggiungiamo un nuovo metodo "blocca" all'oggetto "Pezzo" utilizzando la proprietà "prototype" che viene invocato quando il tetramino attivo non può più scendere e arriva alla fine della griglia di gioco.
    Quando avviene ciò il gioco termina e viene visualizzata l'immagine di "game over" e il suono corrispondente viene riprodotto. La variabile gameOver viene impostata a true per indicare che il gioco è finito. Se però il tetramino non supera la fine del gioco, il colore del quadrato viene aggiunto alla tabella nella posizione (this.y+r, this.x+c).

    Il secondo ciclo for scorre tutte le righe della griglia di gioco. Se una riga è completamente riempita viene eliminata. Le righe sopra la riga eliminata scendono di una posizione e la prima riga viene riempita con il colore "white". Il punteggio viene incrementato di 10 punti per ogni riga eliminata.

    Viene chiamato il metodo disegnaTab per ridisegnare la griglia di gioco aggiornata e il punteggio viene mostrato nell'elemento HTML con l'id EPunteggio.

```
245  document.addEventListener("keydown",CONTROLLA);
246
247  function CONTROLLA(event){
248      if(event.keyCode == 37){
249          p.muoviSinistra();
250          CP = Date.now();
251      }else if(event.keyCode == 38){
252          p.ruota();
253          CP = Date.now();
254      }else if(event.keyCode == 39){
255          p.muoviDestra();
256          CP = Date.now();
257      }else if(event.keyCode == 40){
258          p.muoviGiu();
259      }
260      else if(event.keyCode == 32){
261          p.muoviGiu();
262          p.tuttoGiu();
263      }
264  }
```

20. Registriamo un event listener per l'evento "keydown" che ad ogni pressione di un tasto della tastiera viene chiamata la funzione "CONTROLLA".
    Questa funzione, controlla quale tasto è stato premuto e chiama il metodo corrispondente dell'oggetto "p", che si riferisce al tetramino attivo, per spostare il pezzo.

```
269     let CP = Date.now();
270     let gameOver = false;
271     function rilascia(){
272         let now = Date.now();
273         let delta = now - CP;
274         if(delta > 1000){
275             p.muoviGiu();
276             CP = Date.now();
277         }
278         if(!gameOver){
279             requestAnimationFrame(rilascia);
280         }
281     }
282
283     rilascia();
```

21. Registriamo un event listener per l'evento "keydown" che ad ogni pressione di un tasto della tastiera viene chiamata la funzione "CONTROLLA". Questa funzione, controlla quale tasto è stato premuto e chiama il metodo corrispondente dell'oggetto "p", che si riferisce al tetramino attivo, per spostare il pezzo, La variabile CP tiene traccia dell'ultima volta in cui è stato premuto un tasto e viene utilizzata per controllare la velocità di caduta del blocco corrente. La funzione viene chiamata all'avvio del gioco e, attraverso l'utilizzo della funzione requestAnimationFrame(), controlla continuamente il tempo trascorso e fa cadere il blocco corrente ogni secondo. Questa funzione viene eseguita in modo continuo fino a quando il gioco non finisce. La variabile gameOver viene utilizzata per controllare se il gioco è finito o meno.

```
291 $("#state").click(function(){
292     sottofondo.pause();
293     alert("Gioco in pausa");
294     sottofondo.play();
295 });
296
297 $("#restart").click(function(){
298     sottofondo.pause();
299     if(confirm("Riavviare la partita?")) {
300         window.location.reload();
301     }
302     else sottofondo.play();
303 });
304
305 $("#home").click(function(){
306     sottofondo.pause();
307     if(confirm("Tornare alla pagina principale?")) {
308         $(window).prop("location", "index.html");
309     }
310     else sottofondo.play();
311 });
312
313 window.addEventListener("load", (event) => {
314     sottofondo.setAttribute('src', 'sounds/tetris/soundtrack.mp3');
315     sottofondo.volume = 0.1;
316     sottofondo.play();
317
318 });
319
320 window.onblur = function() {
321     sottofondo.pause();
322 }
323
324 window.onfocus = function() {
325     sottofondo.play();
326 }
```

22. Questa funzione serve per mettere in pausa il gioco usando un banalissimo alert.

23. Questa funzione serve per riavviare la partita solo se alla comparsa del pop-up viene premuto 'OK'.

24. Questa funzione serve per tornare alla pagina principale solo se alla comparsa del pop-up viene premuto 'OK'.

25. Viene implementato un listener degli eventi all'oggetto window che ascolta l'evento load. Quando l'evento load viene scatenato, viene eseguita la funzione contenuta nella freccia, la funzione in se inizializza il file audio e inizia a riprodurlo quando la pagina ha terminato di caricarsi.

26. Questa funzione serve per mettere in pausa la musica quando si esce dalla pagina.

27. Questa funzione serve per riprendere la musica quando si torna sulla pagina.

```
328  $("#music").click(function(){
329      if(song) {
330          sottofondo.pause();
331          song = false;
332        }
333        else {
334          sottofondo.play();
335          song = true;
336        }
337  });
338
339  document.addEventListener("keydown", function(event) {
340      if(event.key === "+"){
341          sottofondo.volume += 0.1;
342      }
343      else if(event.key === "-"){
344          sottofondo.volume -= 0.1;
345      }
346  });
```

28. Questa funzione permette di mettere in pausa o di riprendere la musica, in base al suo stato, tramite l'ausilio di una variabile booleana.

29. Questa funzione permette di alzare o abbassare il volume della musica utilizzando i corrispettivi tasti '+' e '-'.