

## Istruzioni iterative in p5.js

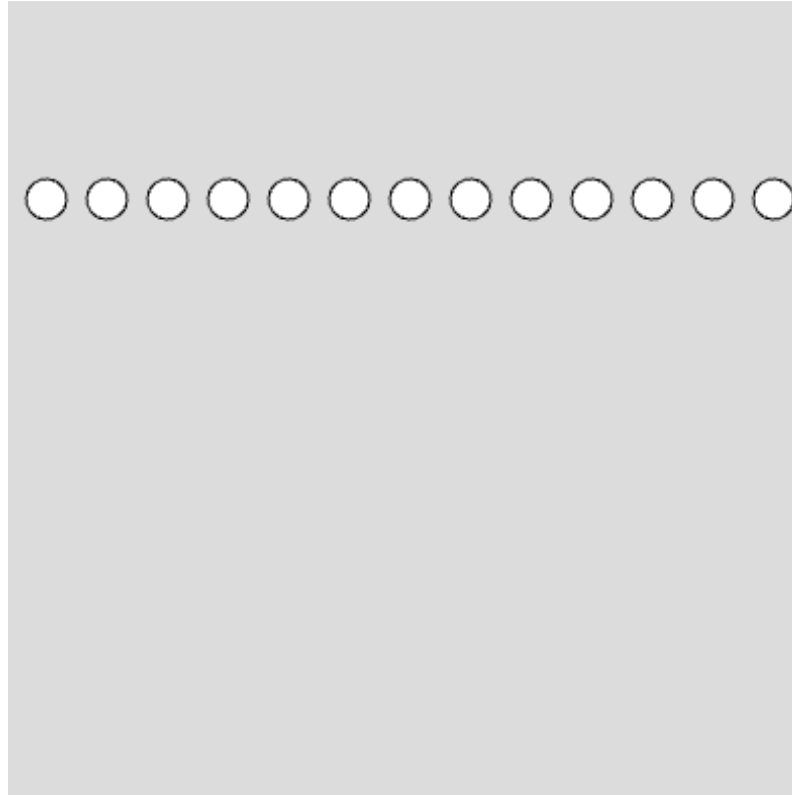
## Perché usare le istruzioni iterative?

Pensa di dover disegnare una serie di cerchi a distanza regolare lungo l'asse x.

```
ellipse(0, 100, 20, 20);  
ellipse(50, 100, 20, 20);  
ellipse(100, 100, 20, 20);  
ellipse(150, 100, 20, 20);  
ellipse(200, 100, 20, 20);  
ellipse(250, 100, 20, 20);
```

- ✗ Non è efficiente nel caso di molti cerchi
- ✗ È difficile da mantenere
- ✗ È difficile da modificare

## Struttura dei cicli While



Finché  $x$  è minore di 400, disegna un cerchio e incrementa  $x$  di 30.

## Struttura dei cicli While (0)

```
let x = 20; // 1. Inizializzazione
function setup() {
  createCanvas(400, 400);
  background(220);
}

function draw() {
  while (x < width) { // 2. Condizione di continuazione
    ellipse(x, 100, 20, 20);
    x = x + 30; // 3. Aggiornamento della variabile
  }
}
```

## Condizione di Uscita

Considerazioni importanti:

- Quando il ciclo deve fermarsi?
- Qual è la condizione limite?
- Come evitare i cicli infiniti?

Example:

```
while (x <= width) { // Interrompi quando x è maggiore o uguale a width
  ellipse(x, 100, 20, 20);
  x = x + 50;
}
```

*provare il codice anche con* `x < width/2`

## Cilci For

Stessi componenti di un ciclo while, ma in una sola riga:

```
for (let x = 0; x < width; x += 50) {  
  ellipse(x, 100, 20, 20);  
}
```

Componenti in una riga:

1. Inizializzazione: `let x = 0`
2. Condizione: `x < width`
3. Aggiornamento: `x += 50`

## Esercizio: pattern di linee diagonali (1)

```
const offset = 10;
const distance = 300;
function setup() {
  createCanvas(600, 400);
  background(235);
}
function draw() {
  for (let i = 0; i < width + distance; i += offset) {
    strokeWeight(1)
    line(i-distance, 0, i, height);
  }
}
```

## Idee per fare pratica

Prova a creare pattern con i cicli:

1. Linee con colori variabili
2. Forme con dimensioni variabili
3. Gradienti di colore



## **Esempio 1. Linee con colori variabili**

Disegna una serie di linee verticali equidistanti con colori casuali

## Soluzione (1b). Linee con colori variabili

```
function setup() {  
  createCanvas(400, 400);  
  background(220);  
  for (let x = 0; x < width; x += 50) {  
    stroke(random(255), random(255), random(255));  
    line(x, 0, x, height);  
  }  
}
```

## **Esempio 2. Forme con dimensioni variabili**

Disegna una serie di cerchi equidistanti con dimensioni casuali  
(con un diametro minimo di 10 e un massimo di 50)

## Soluzione 2. Forme con dimensioni variabili

```
function setup() {  
  createCanvas(400, 400);  
  background(220);  
  for (let x = 0; x < width; x += 50) {  
    let size = random(10, 50);  
    ellipse(x, 100, size, size);  
  }  
}
```

## Esempio 3. Gradienti di colore

Disegna una serie di cerchi equidistanti con un gradiente di colore lungo l'asse x

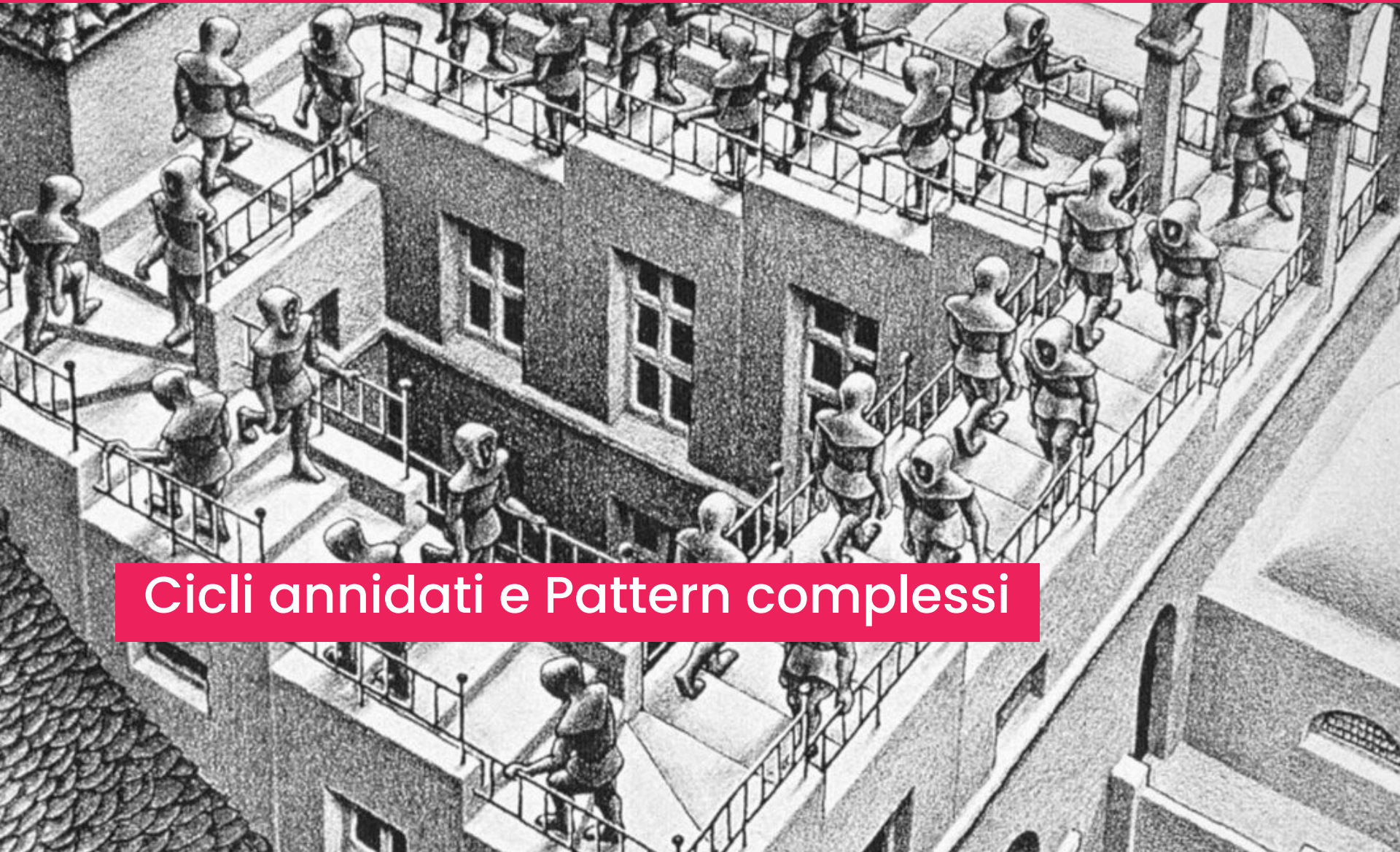
## Soluzione 3. Gradienti di colore

```
function setup() {  
  createCanvas(400, 400);  
  background(220);  
  for (let x = 0; x < width; x += 50) {  
    fill(x, 0, 0);  
    ellipse(x, 100, 20, 20);  
  }  
}
```

## Conclusioni

1. I cicli evitano codice ripetitivo
2. I cicli while sono buoni per il controllo esplicito
3. I cicli for sono preferiti per i pattern di conteggio
4. Considera sempre le condizioni di uscita
5. Usa variabili locali quando possibile

Ricorda: i cicli sono essenziali per creare modelli complessi in modo efficiente!



## Cicli annidati e Pattern complessi



## La funzione draw()

Proviamo a capire come funziona la funzione `draw()` :

```
function draw() {  
  // La funzione draw() viene eseguita continuamente  
  // Immagina di avere un libro che si sfoglia  
  // Ogni esecuzione è come una nuova pagina  
}
```

N.B. Il canvas si aggiorna solo quando `draw()` completa un ciclo!

## Concetti importanti sul Timing

Proviamo a capire come funziona il timing in `draw()` :

```
function draw() {  
  for(let x = 0; x < width; x += 50) {  
    ellipse(x, 100, 20, 20);  
  }  
}
```

- Tutte le forme appaiono **contemporaneamente**
- Il canvas **si aggiorna solo alla fine**, con tutte le forme già disegnate
- Ovvero, **non** vediamo le forme apparire **una alla volta**
- Di **default** il timeFrame è di **60 frame** al secondo

## Introduzione ai Cicli Annidati (1)

In una griglia, abbiamo bisogno di lavorare su due dimensioni (x e y):

```
for(let x = 0; x < width; x += 50) {  
  for(let y = 0; y < height; y += 50) {  
    ellipse(x, y, 20, 20);  
  }  
}
```

Per ogni posizione x:

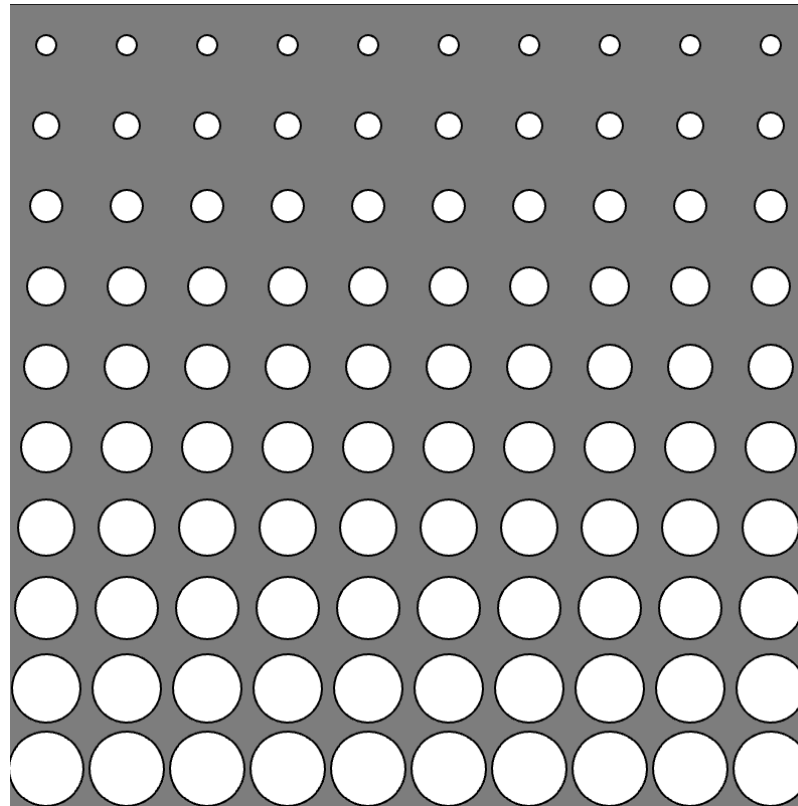
- Disegna cerchi in ogni posizione y
- Crea un modello a griglia completo

## Come funzionano i Cicli Annidati

Proviamo a capire l'ordine di esecuzione:

1. Il ciclo esterno inizia:  $x = 0$
2. Il ciclo interno viene eseguito completamente: da  $y = 0$  a height
3. Il ciclo esterno continua sommando a  $x$  50
4. Il ciclo interno viene eseguito nuovamente: da  $y = 0$  a height
5. Il processo continua fino a completare la griglia

Pensa al processo come "per ogni colonna, riempi tutte le righe"



# Applicazioni dei Cicli Annidati

I cicli annidati sono essenziali per:

- Creare giochi basati su griglia (scacchi, dama)
- Elaborazione di immagini (manipolazione dei pixel)
- Generazione di pattern
- Visualizzazione dei dati
- Mappe basate su tile
- Operazioni matematiche sulle matrici

## Interactive Patterns (1b)

Possiamo rendere i pattern interattivi utilizzando la posizione del mouse all'interno dei cicli:

```
function draw() {  
  for(let x = 0; x < mouseX; x += 50) {  
    for(let y = 0; y < height; y += 50) {  
      ellipse(x, y, 20, 20);  
    }  
  }  
}
```

La griglia ora si espande e si contrae con il movimento del mouse!

## Capire la Dinamicità dei Cicli

Ogni parte del loop può essere dinamica:

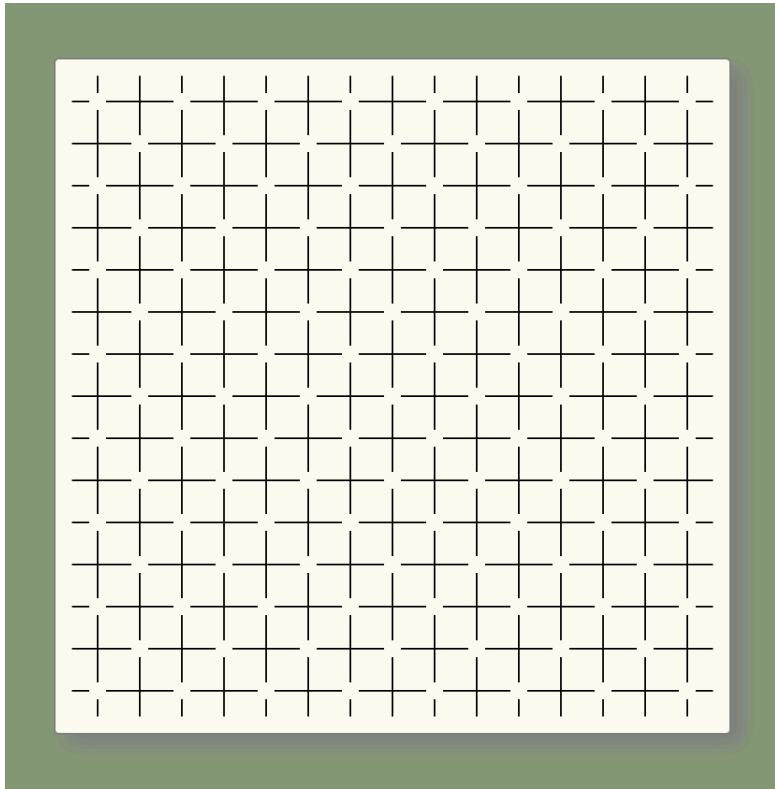
```
// Static loop  
for(let x = 0; x <= width; x += 50)  
  
// Dynamic variation:  
for(let x = 0; x <= mouseX; x += 50)
```

Experiment with different combinations!



## Takeaway Messages

1. Il ciclo `draw()` crea animazioni attraverso i frame
2. I cicli annidati sono perfetti per i pattern basati su griglia
3. Le variabili possono rendere i pattern dinamici
4. Combinare elementi casuali per un aspetto organico
5. L'interazione del mouse aggiunge dinamismo alle griglie
6. Pensare in termini di frame completi



*Karl Reinhardt, 1962*

```
function setup() {
  createCanvas(400, 400);
  background(255, 250, 243);
  //create line grid
  for(var pos = 25; pos < 400; pos+=25){
    stroke(0);
    noFill();
    line(pos,10,pos,390);
    line(10,pos,390,pos);
  }
  //create first grid of dots
  for(var x = 25; x < 400; x += 50){
    for(var y = 25; y < 400; y += 50){
      noStroke();
      fill(255, 250, 243);
      ellipse(x, y, 10);
    }
  }
  //create second grid of dots
  for(var x = 50; x < 400; x += 50){
    for(var y = 50; y < 400; y += 50){
      noStroke();
      fill(255, 250, 243);
      ellipse(x, y, 10);
    }
  }
}
```