Cosa sono le Funzioni?

- Le funzioni sono i **blocchi fondamentali** di JavaScript
- Ci permettono di **organizzare** e **riutilizzare** il codice
- N.B. Abbiamo già usato molte funzioni di p5.js:

```
ellipse(100, 100, 50, 50);
rect(200, 200, 40, 40);
```

Chiamare vs Definire una Funzione

Definire una Funzione

```
function setup() {           // Stiamo DEFINENDO la funzione
    createCanvas(400, 400);
}
```

Chiamare una Funzione

```
ellipse(100, 100, 50, 50);   // Stiamo CHIAMANDO la funzione
```

Funzioni Speciali di p5.js

Funzioni già definite (da p5.js)

- `ellipse()`, `rect()`, `line()`
- `fill()`, `stroke()`, `background()`
- Non dobbiamo definirle, sono parte della libreria

Funzioni che dobbiamo definire noi

- `setup()` - eseguita all'avvio del programma
- `draw()` - eseguita continuamente
- p5.js sa quando eseguirle

Perché usare le Funzioni?

Due motivi principali

1. **Modularità**
    - **Organizzare** il codice in blocchi logici
    - Rendere il programma più **leggibile**
    - Facilitare la **manutenzione**
2. **Riutilizzabilità**
    - Scrivere il codice una volta
    - Usarlo più volte con parametri diversi
    - Evitare la ripetizione del codice

La Sintassi di una Funzione

```
function nomeDellaFunzione() {

    // Il codice della funzione va qui
    // Questo codice verrà eseguito quando
    // la funzione viene chiamata

}
```

Parti importanti:

- La parola chiave `function`
- Il nome della funzione
- Le parentesi tonde `()` per i parametri
- Le parentesi graffe `{}` per il codice della funzione

Una Palla che Rimbalza

```
function draw() {
    background(220);

    // Disegna la palla
    ellipse(ball.x, ball.y, 50, 50);

    // Controlla i rimbalzi
    if (ball.x > width || ball.x < 0) {
        ball.xspeed *= -1;
    }

    // Muovi la palla
    ball.x += ball.xspeed;
}
```

Questo codice può essere diviso in funzioni separate!

Esercizio: Riscrivi il precedente codice utilizzando le funzioni

TODO List:

1. Crea una funzione per **disegnare** la palla
2. Crea una funzione per controllare i **rimbalzi**
3. Crea una funzione per **muovere** la palla

Richiama queste funzioni all'interno di `draw()`

Soluzione

```
function display() {
    ellipse(ball.x, ball.y, 50, 50);
}
function bounce() {
    if (ball.x > width || ball.x < 0) {
        ball.xspeed *= -1;
    }
}
function move() {
    ball.x += ball.xspeed;
}

function draw() {
    background(220);
    display();  // Disegna la palla
    bounce();   // Controlla i rimbalzi
    move();     // Muovi la palla
}
```

`08a_palla_function`

Importante Ricordare

1. Le funzioni devono essere **definite** prima di poter essere **chiamate**
2. Il nome della funzione dovrebbe descrivere ciò che fa
3. Una funzione può essere chiamata più volte
4. Definire una funzione non la esegue - dobbiamo **chiamarla**

Parametri e Argomenti di una Funzione

Il problema:

```
// Disegnare lo stesso elemento più volte...
ellipse(100, 100, 50, 50);
ellipse(200, 100, 50, 50);
ellipse(300, 100, 50, 50);
```

La soluzione:

```
// Creare una funzione e chiamarla più volte!
function disegnaElemento(x, y) {
    ellipse(x, y, 50, 50);
}
```

Parametri e Argomenti

```
// DEFINIZIONE della funzione
function disegnaFiore(x, y, numPetali) {
    // x, y, numPetali sono PARAMETRI
}

// CHIAMATA della funzione
disegnaFiore(200, 300, 8);   // 200, 300, 8 sono ARGOMENTI
```

- I **parametri** sono variabili usate nella definizione
- Gli **argomenti** sono i valori passati quando chiamiamo la funzione

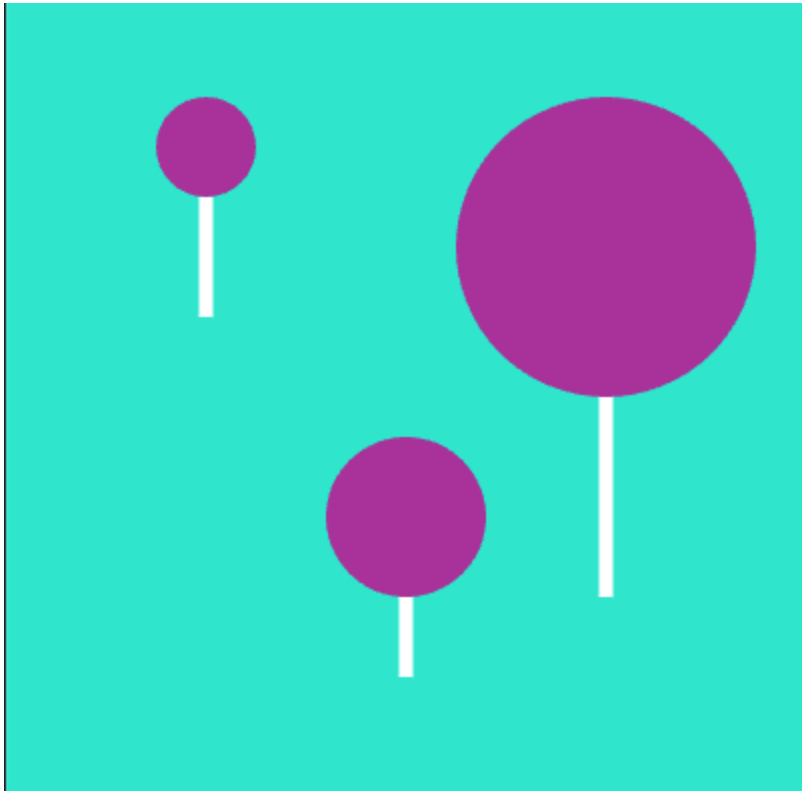
Come Funzionano i Parametri

```
function disegnaFiore(x, y, numPetali) {

    // È come se all'inizio della funzione ci fosse:
    const x = undefined;        // valore dal primo argomento
    const y = undefined;        // valore dal secondo argomento
    const numPetali = undefined;// valore dal terzo argomento

    ellipse(x, y, 20, 20);   // Centro del fiore
    // Codice per disegnare i petali...
}
```

Esempio: Il Lollipop



Esempio: Il Lollipop

```
function lollipop(x, y, diametro, altezza_bastoncino) {
  noStroke();
  fill(255);
  rectMode(CENTER);
  rect(x, y + altezza_bastoncino/2, 7, altezza_bastoncino);

  fill(168, 50, 153);
  ellipse(x, y - diametro/2, diametro);
}

function setup() {
    background(220);
    lollipop(100, 100, 50);  // Lollipop piccolo
    lollipop(300, 200, 150); // Lollipop grande
}
```

`08b_lollipop`

Perché usare i Parametri?

Vantaggi:

1. Lo stesso codice può produrre risultati diversi
2. Rende il codice più flessibile
3. Riduce la ripetizione
4. Rende più facile fare modifiche

Esempio:

```
// Cambiando solo gli argomenti...
lollipop(x, y, 50);      // Lollipop piccolo
lollipop(x, y, 150);     // Lollipop grande
lollipop(x, y, 100);     // Lollipop medio
```

Regole Importanti

1. Il numero di argomenti deve corrispondere al numero di parametri
2. Gli argomenti vengono assegnati ai parametri in ordine
3. I parametri sono variabili locali della funzione
4. I parametri esistono solo all'interno della funzione

Esercizio

Crea una funzione che disegni una **casa** con questi parametri:

- Posizione (x, y)
- Dimensione
- Colore del tetto

`08c_house`

Return Values

Cosa stamperà questo codice?

```
function doppio(x) {
    x * 2;
}

function triplo(x) {
    return x * 3;
}

console.log(doppio(5));   // ???
console.log(triplo(5));   // ???
```

Due Tipi di Funzioni

Funzioni che fanno qualcosa

```
ellipse(100, 100, 50, 50);   // Disegna un cerchio
background(220);             // Colora lo sfondo
```

Funzioni che restituiscono un valore

```
let n = random(100);      // Restituisce un numero casuale
let r = sqrt(25);         // Restituisce 5
```

La Parola Chiave `return`

```
// Una funzione che converte miglia in chilometri
function migliaInKm(miglia) {
    let km = miglia * 1.6;
    return km;  // Restituisce il valore calcolato
}

// Uso della funzione
let maratona = migliaInKm(26.3);
console.log(maratona);  // Stampa: 42.08
```

Come Usare i Valori Restituiti

1. Salvare in una variabile

```
let temperatura = celsiusToFahrenheit(20);
```

2. Usare direttamente in un'altra funzione

```
fill(random(255));   // Usa direttamente il numero casuale
```

3. Usare in una condizione

```
if (distanza > calcolaDistanzaMinima()) {
    // fai qualcosa
}
```

Esempio: Conversione di Temperature

```
function celsiusToFahrenheit(celsius) {
    // Formula: (°C × 9/5) + 32 = °F
    let fahrenheit = (celsius * 9/5) + 32;
    return fahrenheit;
}

// Proviamo alcune temperature
console.log(celsiusToFahrenheit(0));   // 32°F
console.log(celsiusToFahrenheit(20));  // 68°F
console.log(celsiusToFahrenheit(37));  // 98.6°F
```

Esempio: Calcolo della Distanza

```
function calcolaDistanza(x1, y1, x2, y2) {
    // Teorema di Pitagora
    const diffX = x2 - x1;
    const diffY = y2 - y1;
    const distanza = sqrt(diffX * diffX + diffY * diffY);
    return distanza;
}
// Uso nella funzione draw
function draw() {
    let d = calcolaDistanza(mouseX, mouseY, width/2, height/2);
    if (d < 50) {
        fill(255, 0, 0);
    }
}
```

`08d_distanza`

N.B. Il Valore di Ritorno è Singolo

❌
```
// Non possiamo restituire più valori direttamente
function calcolaDimensioni() {
    return larghezza, altezza;  // Non funziona!
}
```

✅
```
// Possiamo restituire un oggetto
function calcolaDimensioni() {
    return {
        larghezza: 100,
        altezza: 50
    };
}
```

🏠Esercizi

1. Conversione di Unità

Scrivi una funzione che converta:

- Centimetri in pollici (1 inch = 2.54 cm)
- Euro in dollari (cerca il tasso di cambio!)

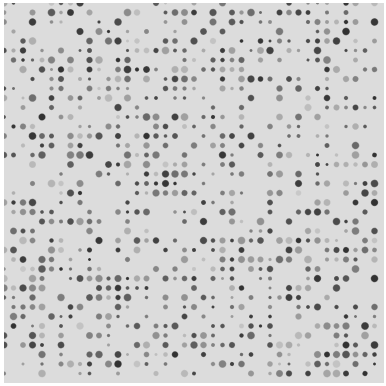
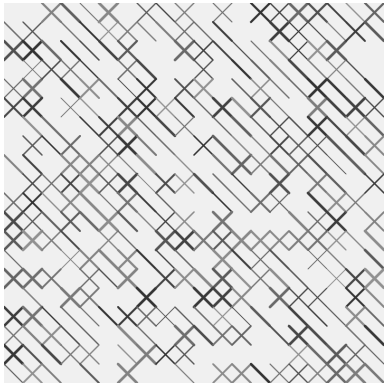
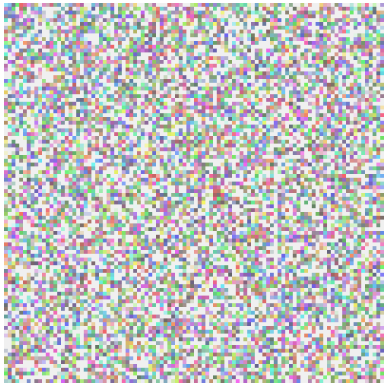
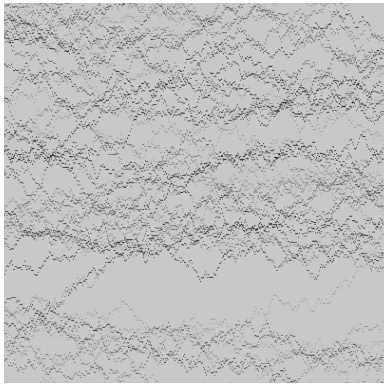
2. Calcoli Geometrici

Scrivi funzioni che calcolino:

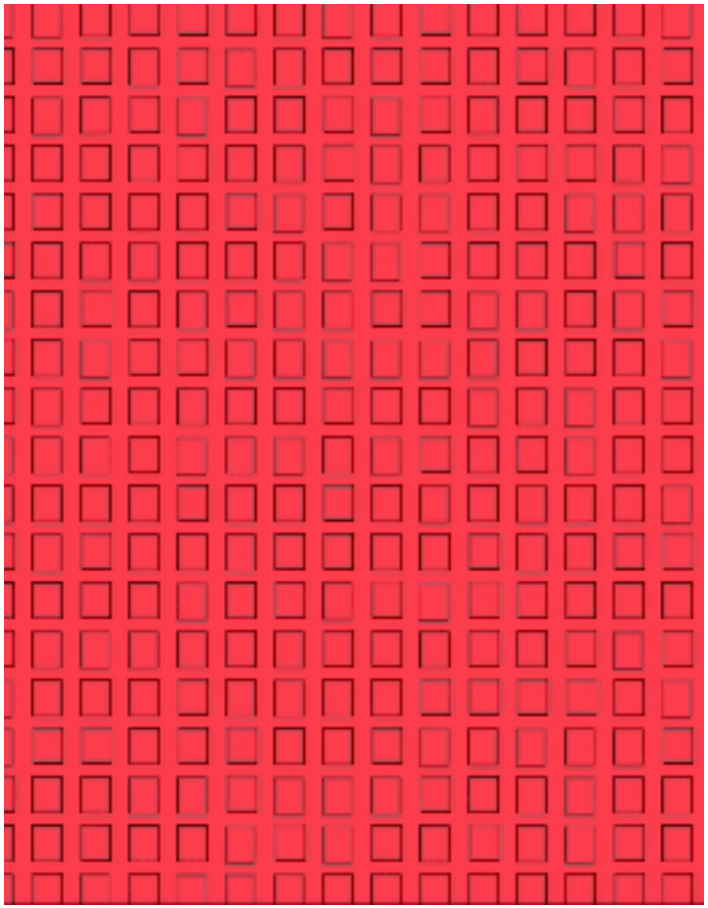
- L'area di un triangolo
- Il perimetro di un cerchio

3. Converti in funzioni

Riscrivi il codice degli esercizi precedenti utilizzando le funzioni

4. Scrivi un testo che spiega come faresti questo video



Istruzioni per scrivere le istruzioni

- Analizza il video e identifica gli elementi principali
- Per ogni elemento osserva stile, posizione, movimento
- Identifica cosa è unico e cosa invece è variazione di un pattern
- Identifica eventuali interazioni dell'utente
- Non usare codice per descrivere il video!!!