# FUNKTIONEN

## Allgemeines

= Prozeduren, Methoden, Subroutine

Der Programmcode wird in kleine Unterprogramme untergliedert.

Modularität

> Erleichtert das Lesen

> Erleichtert das Überarbeiten

# FUNKTIONEN

Allgemeines

= Prozeduren, Methoden, Subroutine

Der Programmcode wird in kleine Unterprogramme untergliedert.

Modularität

> Erleichtert das Lesen

> Erleichtert das Überarbeiten

Wiederverwendbarkeit

> Funktionen können beliebig oft aufgerufen & benutzt werden

```
void draw ()
{
        background (255);
        drawRects ();
        drawEllipses ();
        drawLines ();
}
```

# FUNKTIONEN

## Aufbau

### Funktionen Definieren

```
returnType funktionsName (Parameter)
{
        //Code
}
```

# FUNKTIONEN

## Aufbau

### Funktionen Definieren

```
returnType funktionsName (Parameter)
{
        //Code
}

void drawEllipses ()
{
        for (int i = 0; i < 10; i++)
        {
        float s = random (10, 50);
        ellipse (random(width), random (height), s, s);
        }
}
```

# FUNKTIONEN

## Aufbau

### Funktionen Definieren

```
returnType funktionsName (Parameter)
{
        //Code
}
```

```
void drawEllipses ()
{
        for (int i = 0; i < 10; i++)
        {
        float s = random (10, 50);
        ellipse (random(width), random (height), s, s);
        }
}
```

```
void draw ()
{
        background (255);
}
```

# FUNKTIONEN

## Aufbau

Funktionen Definieren (immer außerhalb von anderen Funktionen)

Funktionen müssen aufgerufen werden, damit sie ausgeführt werden

```
returnType funktionsName (Parameter)
{
        //Code
}
```

```
void drawEllipses ()
{
        for (int i = 0; i < 10; i++)
        {
        float s = random (10, 50);
        ellipse (random(width), random (height), s, s);
        }
}
```

```
void draw ()
{
        background (255);
        drawEllipses();
}
```

## Funktionen mit Parametern

Parameter sind Werte, die der Funktion mitgegeben werden

```
returnType funktionsName (Parameter)
{
        //Code
}
```

```
void drawEllipses (int n)
{
        for (int i = 0; i < n; i++)
        {
        float s = random (10, 50);
        ellipse (random(width), random (height), s, s);
        }
}
```

```
void draw ()
{
        background (255);
        drawEllipses(50);
}
```

## Funktionen mit Parametern

Parameter sind Werte, die der Funktion mitgegeben werden

Bei mehreren Paramtern: Trennung mit Komma

```
returnType funktionsName (Parameter)
{
        //Code
}
```

```
void drawEllipses (int n, float minS, float maxS)
{
        for (int i = 0; i < n; i++)
        {
        float s = random (minS, maxS);
        ellipse (random(width), random (height), s, s);
        }
}
```

```
void draw ()
{
        background (255);
        drawEllipses(50, 10.5, 45.87);
}
```

## Funktionen mit Parametern

Parameter sind Werte, die der Funktion mitgegeben werden

Bei mehreren Paramtern: Trennung mit Komma

Die übergebenen Paramter müssen vom selben Datentyp sein, wie in der Funktion definiert wurden

```
drawEllipses(50, 10.5, 45.87);
```

```
void drawEllipses (int n, float minS, float maxS)
{
    for (int i = 0; i < n; i++)
    {
    float s = random (minS, maxS);
    ellipse (random(width), random (height), s, s);
    }
}
```

```
void draw ()
{
    background (255);
    drawEllipses(50, 10.5, 45.87);
}
```

## Funktionen mit Parametern

Wir haben bereits jede Menge vordefinierte Funktionen (mit und ohne Paramtern) kennengelernt.

```
void draw ()
{
        noStroke();
        ellipse (10, 20, 50, 80);
}
```

```
void ellipse (float x, float y, float w, float h)
{
        // Ellipse wird gezeichnet
}

void noStroke ()
{
        // Kontur wird unsichtbar
}
```

## Funktionen mit Rückgabewert

Das erste Schlüsselwort bei der Funktionsdefinition gibt den Rückgabewert an

## Funktionen mit Rückgabewert

Das erste Schlüsselwort bei der Funktionsdefinition gibt den Rückgabewert an

void = Nichts = die Funktion gibt nichts zurück

```
returnType funktionsName (Parameter)
{
    //Code
}
```

```
void draw ()
{
    //Code
}
```

## Funktionen mit Rückgabewert

Das erste Schlüsselwort bei der Funktionsdefinition gibt den Rückgabewert an

void = Nichts = die Funktion gibt nichts zurück

Funktionen haben meist den Zweck komplexe Berechnungen auszuführen. Dafür können Funktionen mit einem Rückgabewert eingesetzt werden.

```
int summe (int a, int b, int c)
{
        int total = a+b+c;
        return total;
}
```

```
void draw ()
{
        int berechnung = summe (2, 4, 6);
}
```

## Funktionen mit Rückgabewert

Das erste Schlüsselwort bei der Funktionsdefinition gibt den Rückgabewert an

void = Nichts = die Funktion gibt nichts zurück

Funktionen haben meist den Zweck komplexe Berechnungen auszuführen. Dafür können Funktionen mit einem Rückgabewert eingesetzt werden.

Funktionen mit Rückgabewert müssen immer einen Wert zurückgeben. Der Wert muss vom selben Datentyp sein wie vor dem Funktionsname angegeben wurde.

```
int summe (int a, int b, int c)
{
        int total = a+b+c;
        return total;
}
```

```
void draw ()
{
        int berechnung = summe (2, 4, 6);
}
```

## Funktionen mit Rückgabewert

Das erste Schlüsselwort bei der Funktionsdefinition gibt den Rückgabewert an
void = Nichts = die Funktion gibt nichts zurück
Funktionen haben meist den Zweck komplexe Berechnungen auszuführen. Dafür können Funktionen mit einem Rückgabewert eingesetzt werden.
Funktionen mit Rückgabewert müssen immer einen Wert zurückgeben. Der Wert muss vom selben Datentyp sein wie vor dem Funktionsname angegeben wurde.

```
float random (float min, float max)
{
      // Zufallszahl wird erzeugt
      return zufallszahl;
}
```

```
void draw ()
{
      float zufallszahl = random (0.5, 20);
}
```

# EXKURS: DEFINITION UND AUSFÜHREN

DEFINTION

Funktionen

```
void drawTenEllipses ()
{
    for (int i = 0; i < 10; i++)
    {
    ellipse (random (width), random (height), 10, 10);
    }
}
```

Variablen

```
float x;
float y;

x = 10;
y = 20;
```

>>>> bis hier ist noch nichts passiert <<<<

# EXKURS: DEFINITION UND AUSFÜHREN

DEFINTION

Funktionen

```
void drawTenEllipses ()
{
    for (int i = 0; i < 10; i++)
    {
    ellipse (random (width), random (height), 10, 10);
    }
}
```

Variablen

```
float x;
float y;

x = 10;
y = 20;
```

>>>> bis hier ist noch nichts passiert <<<<

AUSFÜHREN

```
void draw ()
{
    drawTenEllipses();
}
```

```
void draw ()
{
    ellipse (x, y, 10, 10);
}
```