UI

Diese Methoden müssen von `JComponent`-Elementen aufgerufen werden, z. B. `label.setValue("Text");`.

Alle UI-Komponenten (`JComponent`)

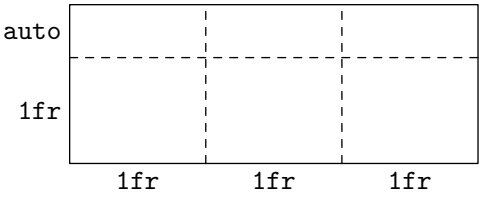
- `setValue(String)`
  Ändert den Text/Wert des Elements.
- `getValue(): String`
  Liefert den Text/Wert des Elements zurück.
- `show()/hide()`
  Macht die Komponente sichtbar / unsichtbar.
- `setStyle(String css-eigenschaft, String wert)`
  Legt eine CSS-Eigenschaft fest, z. B. `setStyle("color","red")`
- `setStyle(String css-eigenschaft, String wert)`
  Legt eine CSS-Eigenschaft fest, z. B. `setStyle("color","red")`
- `setEnabled(boolean)`
  Aktiviert (`true`) bzw. deaktiviert (`false`) die Komponente.
- `setActionCommand(String)`
  Legt das `ActionCommand` dieser Komponente fest.
- `getActionCommand(): String`
  Gibt das `ActionCommand` dieser Komponente zurück. Falls keines definiert wurde, wird der Text der Komponente zurückgegeben.
- `addEventListener(String event, (ev)->{Anweisungen})`
  Legt fest, welche Anweisungen ausgeführt werden sollen, wenn ein bestimmtes Ereignis eintritt. Der Parameter `ev` enthält Infos zu dem Event. Mögliche Events sind z. B.:

`"click"`	Komponente wird angeklickt.
`"pointerover"`	Maus betritt die Komponente.
`"pointerout"`	Maus bewegt sich von der Komponente weg.
`"pointermove"`	Maus bewegt sich über der Komponente.
`"pointerdown"`	Maustaste wird gedrückt über der Komponente.
`"pointerup"`	Maustaste wird losgelassen über der Komponente.
`"change"`	Der Wert der Komponente ändert sich.
`"input"`	Es wird etwas in die Komponente eingegeben.

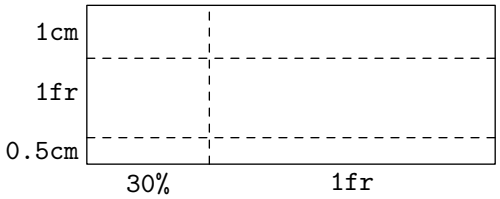
- `setX(double)/setY(double)`
  Legt die x/y-Koordinate des Mittelpunkts fest. Klappt nur, wenn sich das Element in einem `Canvas` befindet.
- `setWidth(double)/setHeight(double)`
  Legt die Breite / Höhe fest. Klappt nur, wenn sich das Element in einem `Canvas` befindet.
- `changeX(double)/...Y/...Width/...Height`
  Ändert den entsprechenden Wert um die Angabe. Klappt nur, wenn sich das Element in einem `Canvas` befindet.

Container (`JPanel`, `JFrame` und `Canvas`)

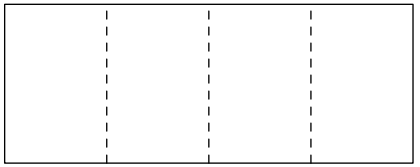
- `setLayout(String)`
  Legt das **Layout** fest. Dieses bestimmt, wie die Kinder angeordnet werden. Syntax: `Höhe Höhe ... Höhe / Breite Breite ... Breite`.



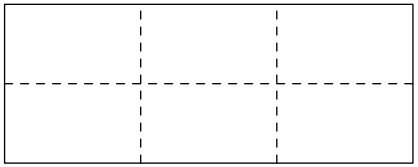
`setLayout("auto 1fr / 1fr 1fr 1fr")`



`setLayout("1cm 1fr 0.5cm / 30% 1fr")`



`setLayout("4")`



`setLayout("3/2")`

- `add(JComponent)`
  Fügt die Komponente hinzu.
- `remove(JComponent)`
  Entfernt die Komponente.
- `getChild(int)`
  Gibt die Kind-Komponente an der Stelle zurück.
- `getChildCount()`
  Die Anzahl der Kind-Komponenten.

Grafiken mit `Canvas`

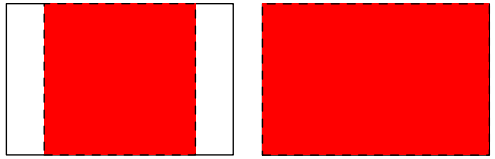
Alle Methoden müssen von einem `Canvas` ausgeführt werden.

Zeichnen-Befehle

- `clear()`
  Löscht alle Zeichnungen.
- `drawLine(double x1,y1,x2,y2)`
  Linie von (x1 | y1) zu (x2 | y2)
- `drawCircle(double mx,my,r)` / `fillCircle(double mx,my,r)`
  Kreis mit Mittelpunkt (mx | my) und Radius `r`
- `drawRect(double mx,my,w,h)` / `fillRect(double mx,my,w,h)`
  Rechteck mit Mittelpunkt (mx | my), Breite `w`, Höhe `h`
- `write(String t, double mx, double my, String align)`
  Schreibt den Text `t` an den Punkt (mx | my). Der optionale Parameter `align` bestimmt die Ausrichtung:
  - `left/right/center`: Links/rechts oder in der Mitte.
  - `top/bottom/middle`: Oben/unten oder in der Mitte.
- `drawImage(String url, double mx,my,w,h,winkel, boolean mirror)`
  Zeichnet das Bild in das Rechteck mit Mittelpunkt (mx | my), Breite `w`, Höhe `h`, gedreht um `winkel` und gespiegelt, wenn `mirror==true`.

Canvas-Einstellungen

- `setAxisX(double min, double max)`
  Legt fest, bei welchem x-Wert der Canvas startet und bis zu welchem x-Wert er geht.
- `setAxisY(double min, double max)`
  Legt fest, bei welchem y-Wert der Canvas startet und bis zu welchem y-Wert er geht.
- `setSizePolicy(String)`
  Legt fest, ob sich der Canvas ausdehnen soll (`"stretch"`) oder ob er eingepasst werden soll (`"fit"`).



Zeichnen-Einstellungen

Jeder dieser Befehle beeinflusst alle **nachfolgenden Zeichnen-Befehle**.

- `setColor(String)`
  Legt die Farbe fest, z. B. `setColor("red");` oder `setColor("#FF69B4");`
- `setOpacity(double)`
  Legt die Transparenz fest von 0.0 (unsichtbar) über 0.5 (halb durchscheinend) bis 1.0 (keine Transparenz).
- `setLinewidth(double)`
  Legt Liniendicke fest.
- `setFontsize(double)`
  Legt die Schriftgröße fest.
- `setFont(String)`
  Legt die Schriftart fest, z. B. `setFont("Arial");`
- `setMirrored(boolean)`
  `setMirrored(true);` ⇒ alle Zeichnungen spiegeln.
- `rotate(double w, double mx, double my)`
  Dreht die Zeichnung um den Winkel `w` um den Mittelpunkt (mx | my).
- `scale(double sx, double sy, double mx, double my)`
  Vergrößert/verkleinert die Zeichung vom Mittelpunkt (mx | my) aus mit dem Vergrößerungsfaktor `sx` in x-Richtung und `sy` in y-Richtung.
- `translate(double dx, double dy)`
  Verschiebt die Zeichnung um `dx` in x-Richtung und um `dy` in y-Richtung.
- `setTransform(double m00, double m10, double m01, double m11, double m02, double m12)`
  Legt die Transformationsmatrix fest auf
  $$\begin{pmatrix} m_{00} & m_{01} & m_{02} \\ m_{10} & m_{11} & m_{12} \\ 0 & 0 & 1 \end{pmatrix}$$
- `reset()`
  Setzt alle Eigenschaften auf die Ursprungswerte zurück.
- `save()`
  Speichert die aktuellen Eigenschaften des Canvas.
- `restore()`
  Stellt die zuletzt gespeicherten Eigenschaften des Canvas wieder her.

Assets

- Assets sind statische Dateien wie Bilder, Sounds, Videos usw.
- Assets werden in JavaApp hochgeladen (Projekt → Assets).
- Um ein Asset in HTML oder CSS zu verwenden, muss man `asset(name)` schreiben, wobei `name` der Name des Assets in JavaApp ist:

```
<img src="asset(bild)"/>
```

Sound

Erst MP3-Datei als Asset hochladen und Namen geben.

⊳ `new Sound(String asset)`
Erzeugt ein neues `Sound`-Objekt. Der Parameter ist der Name der entsprechenden Datei, die als Asset hochgeladen wurde.

⊳ `play(boolean)`
Startet die Wiedergabe (ggf. neu). Falls der Parameter `true` ist, in Endlosschleife.

⊳ `pause()`
Pausiert die Wiedergabe.

⊳ `resume()`
Beendet die Pausierung.

⊳ `stop()`
Beendet die Wiedergabe und setzt den Sound auf Anfang zurück.

⊳ `setSource(String)`
Legt fest, welche Datei abgespielt werden soll.

⊳ `getDuration(): int`
Liefert die Wiedergabezeit des Sounds in Millisekunden zurück.

⊳ `getCurrentTime(): int`
Liefert die Wiedergabeposition zurück (in Millisekunden).

⊳ `isEnded(): boolean`
Liefert `true`, falls die Wiedergabe beendet ist, ansonsten `false`.

⊳ `static beep(int frequency, double volume, int duration)`
Spielt einen Ton ab. Bsp: `Sound.beep(440,1,1000);`

JImage

UI-Komponente, die ein Bild darstellt. Erst PNG oder JPG als Asset hochladen und Namen geben.

⊳ `new JImage(String asset)`
Erzeugt ein neues `JImage`-Objekt.

Gamepad

Jedes Gamepad-Objekt erzeugt ein On-Screen-Gamepad, das automatisch mit einem physischen Gamepad verbunden wird, falls ein solches Gamepad an das Gerät angeschlossen wird.

⊳ `new Gamepad()`
Erzeugt ein neues Gamepad und zeigt es auf dem Bildschirm an.

⊳ `setEventListener(String button, String event, (ev)->{Anweisungen})`
Legt fest, was passieren soll, wenn ein Button gedrückt bzw. losgelassen wird.
- `button`: `"A"`, `"B"`, `"X"`, `"Y"`, `"LEFT"`, `"RIGHT"`, `"UP"` oder `"DOWN"`.
- `event`: `"press"` oder `"release"` oder `"click"`

⊳ `setPosition(String left, String bottom)`
Legt den Abstand zum linken unteren Bildschirmrand fest.
Z. B. `setPosition("2cm","3cm")`

⊳ `setWidth(String width)`
Legt die Breite des Gamepads inklusive Padding fest.
Z. B. `setWidth("50%")`

⊳ `setPadding(String p)`
Legt fest, wie viel Abstand das Steuerkreuz und die ActionButtons vom Rand haben sollen. Z. B. `setPadding("0.5cm")`

⊳ `getDirection(): String`
Liefert einen String zurück, der die Richtung beschreibt, in die das Steuerkreuz gerade zeigt: `"n"`, `"s"`, `"w"`, `"e"`, `"nw"`, `"ne"`, `"sw"` oder `"se"`.

⊳ `isLeftPressed()/isRightPressed()/isUpPressed()/isDownPressed(): boolea`

Liefert `true` zurück, wenn der entsprechende Richtungsbutton gedrückt ist, sonst `false`.

Konsole

- `System.out.println(String)`
  Gibt eine Zeile in der Konsole aus.
- `System.in.read()`
  Wartet auf einen Tastendruck/Klick des Users.
- `System.console().readLine(): String`
  Wartet darauf, dass der User etwas eingibt. Gibt die Eingabe zurück.
- `System.clear()`
  Löscht die gesamte Konsole.