PostScript Cheatsheet

A quick reference guide to the PostScript page description language, covering fundamental concepts, operators, and syntax for generating graphics and text.

Core Concepts

Stack Operations

PostScript is a stack-based language. Most operators take their arguments from the stack and place their results back onto the stack.

- `push` - Places an operand on the stack.
- `pop` - Removes the top operand from the stack.
- `exch` - Exchanges the top two operands on the stack.
- `dup` - Duplicates the top operand.
- `clear` - Empties the stack.

Example:

```
1 2 3  % pushes 1, 2, and 3 onto the stack
```

Data Types

`integer`	Whole numbers (e.g., `1`, `-10`, `0`).
`real`	Floating-point numbers (e.g., `3.14`, `-0.5`).
`boolean`	`true` or `false`.
`string`	Sequence of characters enclosed in parentheses (e.g., `(Hello, world!)`).
`array`	Ordered collection of objects enclosed in square brackets (e.g., `[1 2 3]`).
`dictionary`	Collection of key-value pairs.

Variables and Procedures

- `define` - Associates a name with a value or procedure.
- Variables are declared using `/name value def`.
- Procedures are defined using `{ ... } def`.

Example:

```
/myvar 10 def  % Defines a variable named
myvar with value 10
/square {dup mul} def % Defines a procedure
named square
myvar square % calls procedure
```

Graphics Operations

Path Construction

`newpath`	Starts a new path.
`moveto`	Moves the current point to (x, y). Usage: `x y moveto`.
`lineto`	Draws a line from the current point to (x, y). Usage: `x y lineto`.
`curveto`	Draws a Bezier curve. Usage: `x1 y1 x2 y2 x3 y3 curveto` (control points 1 & 2, endpoint 3).
`closepath`	Closes the current path by drawing a line to the starting point.

Drawing and Filling

`stroke`	Draws a line along the current path.
`fill`	Fills the area enclosed by the current path.
`eofill`	Fills the path using the even-odd rule.

Color and Line Attributes

- `setrgbcolor` - Sets the current color using RGB values. Usage: `red green blue setrgbcolor` (values between 0 and 1).
- `setgray` - Sets the current color using a grayscale value. Usage: `gray setgray` (value between 0 and 1).
- `setlinewidth` - Sets the line width. Usage: `width setlinewidth`.
- `setlinecap` - Sets the line cap style (0 = butt, 1 = round, 2 = square).
- `setlinejoin` - Sets the line join style (0 = miter, 1 = round, 2 = bevel).

Example:

```
0.5 0.2 0.8 setrgbcolor % sets the color to a
shade of purple
2 setlinewidth % sets line width to 2 points
```

Text Operations

Font Handling

`findfont`	Loads a font. Usage: `(FontName) findfont`.
`scalefont`	Scales the font. Usage: `font size scalefont`.
`setfont`	Sets the current font. Usage: `font setfont`.

Text Display

`show`	Displays a string at the current position. Usage: `(text) show`.
`stringwidth`	Calculates the width of a string. Returns width and height. Usage: `(text) stringwidth`.
`showpage`	Displays the current page and resets the graphics state.

Text Positioning

Text positioning is done via `moveto`. Ensure a font is set before displaying text.

Example:

```
/Helvetica findfont 12 scalefont setfont
100 500 moveto
(Hello, PostScript!) show
```

Control Structures

Conditional Statements

`if`	Executes a code block if a condition is true. Usage: `boolean {code} if`.
`ifelse`	Executes one code block if a condition is true, and another if false. Usage: `boolean {true_code} {false_code} ifelse`.

Looping

`repeat`	Executes a code block a specified number of times. Usage: `count {code} repeat`.
`loop`	Executes a code block indefinitely until explicitly stopped with `exit`.
`for`	Executes a code block for a range of values. Usage: `initial increment limit {code} for`.

Example

```
/n 5 def
0 1 n {
  dup mul  % Square the number
  ==       % Print the squared value
} for
```

This example calculates and prints the square of numbers from 1 to 5.