basil.js / Cheatsheet

basil.js

v0.3

Cheatsheet by Ludwig Zeller.
www.basiljs.ch

Color

```
b.color(255,0,0); // creates a color in RGB, also b.color(255) for grayscale
b.fill(255,0,0); // same as b.color(), but sets the color as fill color
b.noFill(); // turns off filling in all following commands
b.noStroke(); // turns off strokes in all following commands
b.stroke(255); // sets the stroke color
```

Drawing

```
b.ellipse(x,y,width,height); // draws an ellipse
b.line(x1,y1,x2,y2); // draws a line from one point to another
b.rect(x,y,width,height); // like b.ellipse() but for rectangles
b.ellipseMode(b.CORNER); // b.CORNER, b.CORNERS, b.CENTER
b.rectMode(b.CENTER); // b.CORNER, b.CORNERS, b.CENTER
b.strokeWeight(0.3); // set the strokes in the current unit
```

Transformation

```
b.itemX(obj, x); // positions the given object on the x axis
b.itemY(obj, y); // positions the given object on the y axis
b.itemWidth(obj, width); // sets the width for the given object
b.itemHeight(obj, height); // sets the height for the given object
b.itemPosition(obj, x, y); // repositions the given object
b.itemSize(obj, width, height); // resizes the given object
b.bounds(obj); // returns an object with width, height, left, right, top, bottom
```

and for text: baseline, xHeight as fields to describe the dimensions of the given object

```
b.height // the height of the page, use b.height / 2 to center vertically
b.width // the width of the page, use b.width/ 2 to center horizontically
```

Typography

```
b.stories(obj); // returns an array with all stories of obj
b.paragraphs(obj); // returns an array with all paragraphs of obj
b.lines(obj); // returns an array with all lines of obj
b.words(obj); // returns an array with all words of obj
b.characters(obj); // returns an array with all characters of obj
b.objectStyle("myStyle"); // return or create given style
b.characterStyle("myStyle"); // return or create given style
b.paragraphStyle("myStyle"); // return or create given style
b.text(txt,x,y,width,height); // create a textframe
b.textAlign(Justification.LEFT_ALIGN,
VerticalJustification.BOTTOM_ALIGN); // check Jongware
b.textFont("Helvetica","Bold"); // set default font
b.textKerning(70); // set default kerning
b.textLeading(96); // set default leading
b.textSize(72); // set default font size
b.textTracking(20); // set default tracking
b.typo(obj, "pointSize", 48); // change special parameters,
```

check Jongware and separate typography cheatsheet for more information about available fields. obj can be a Document, Spread, Page, Layer, Story, TextFrame or any Text object like Line, Paragraph, Word or Character.

Meta

```
b.doc(); // returns the current document
b.page(5); // sets the current page
b.layer("generatedStuff"); // sets the layer to draw on
b.units(b.MM); // set the unit system to b.MM, b.IN, b.CM, b.PX, b.PT
b.guideX(200); // creates a new guide at the given position, also b.guideY();
b.selection(); // returns the first selected object
b.selections(); // returns an array with all selected objects of the document
b.label("myLabel"); // returns the first object with that script label
b.labels("myLabel"); // returns objects with that script label
b.go(); // starts basil.js ...
b.close(); // closes the currently focused document
```

Basics

```
var myNumber = 5; // Integer numbers...
var myNumber = 3.5712; // Floating-point precision numbers...
var myString = "Hello Basil"; // Strings...
var myChar = 'c'; // Characters...
var myBoolean = true; // ... and Booleans are all stored to the "var" type

// This is an inline comment
//~ You can toggle this one with Shift+Cmd+k
/* This is a block comment,
that can go over several lines */

function myFunction (param1, param2){
  // code within the brackets has access to param1 and param2
}
myFunction(234, "Some String"); // this is how you call functions
```

Control Structures

```
// for loops
for( var i = 0; i < 100; i++  )
{
  println( i ); // prints 100 numbers counting upwards
}

// if else conditionals
if ( a > 3 )
{ ... } // execute if a is bigger than 3
else { ... } // execute if a is smaller or equal to 3

// if else conditionals
a == b // equal
a === b // equal value and of the same data type... to be VERY sure
a != b // not equal
a > b // bigger than
a < b // smaller than
a >= b // bigger or equal
a <= b // smaller or equal

// You can combine statements with logical AND "&&" resp. the logical OR "||"
if (  a == b && b == c ) { ... } // AND
if (  a == b || b == c ) { ... } // OR
```

Math

```
b.floor(1.324); // rounds the given value down to 1
b.map(val,0,1,2,5); // maps val between 2-5 as it moves between 0-1
+, -, *, / // addition, substraction, multiplication and division
foo = foo + 5; // increases the variable foo by 5
foo += 5; // a short version of foo = foo + 5;
foo++; // even shorter version for foo = foo + 1;
b.noise(x,y,z); // access a precomputed 1, 2 or 3D noise space
b.random(50); // total chaos between 0-50.
```

Image

```
b.image(filename,x,y,w,h); // draw a picture from the data folder
b.transformImage(img,x,y,w,h); // resizing pictures...
b.imageMode(); // choose between b.CORNER, b.CORNERS and b.CENTER
```

Output

```
b.println("Hello Basil"); // prints the message to the console
b.inspect(obj); // recursive print for complex types like objects and arrays
b.savePDF("export.pdf", true); // outputs a pdf next to the
```

InDesign document file. The second parameter sets if the export dialogue should pop up. If false, the last used settings will be used again.