

# Deck



a Go package for presentations

---

## DECK: a package for presentations

Deck is a package written in Go

That uses a singular markup language

With elements for text, lists, code, and graphics

All layout and sizes are expressed as percentages

Clients are interactive or create formats like PDF or SVG

Elements

# Hello, World

A block of text, word-wrapped to a specified width.  
You may specify size, font, color, and opacity.

```
package main  
  
import "fmt"  
  
func main() {  
    fmt.Println("Hello, World")  
}
```

<text>...</text>

## bullet

- Point A
- Point B
- Point C
- Point D

## plain

First item

Second item

The third item

the last thing

## number

1. This
2. That
3. The other
4. One more

```
<list>...</list>
```

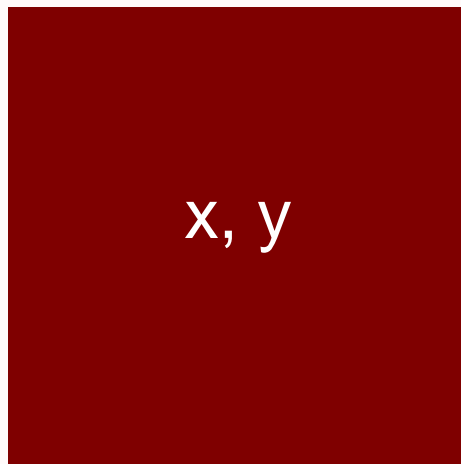
height



width

```
<image ... />
```

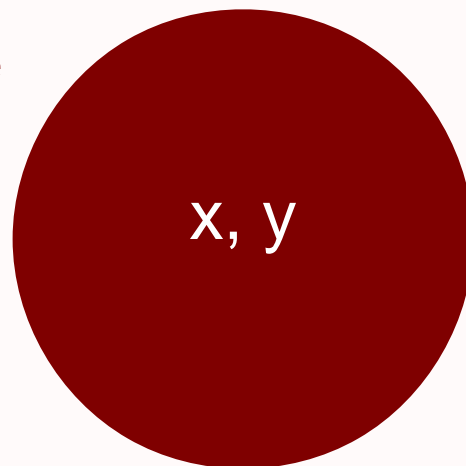
height (relative  
to element  
or canvas  
width)



width

```
<rect ... />
```

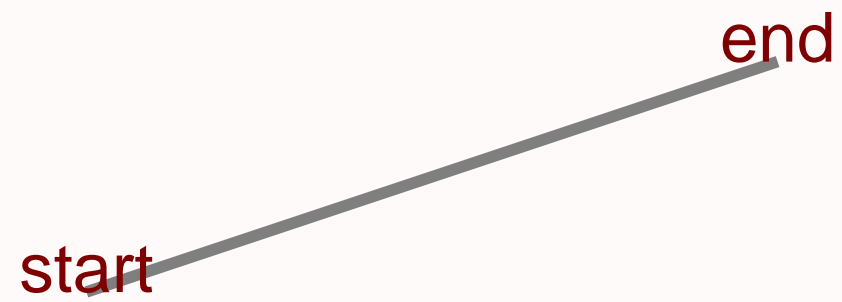
height (relative  
to element  
or canvas  
width)



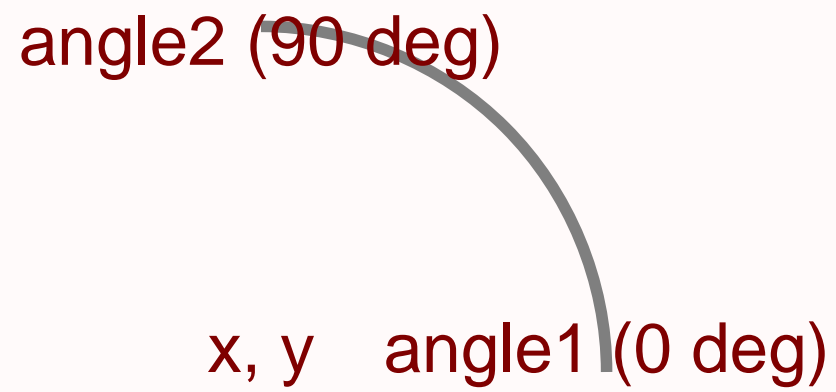
width

```
<ellipse ... />
```





```
<line .../>
```



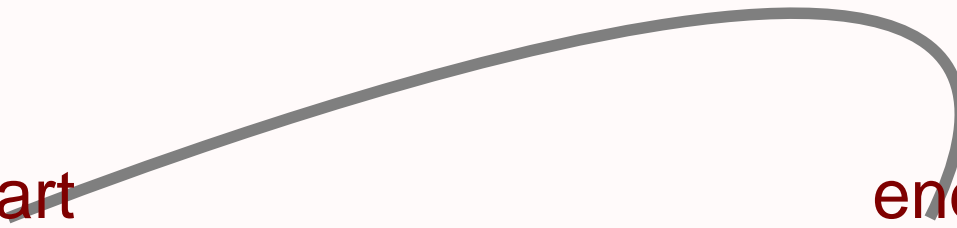
`<arc ... />`

control

start

end

<curve ... />



# Markup and Layout

|                         |  |
|-------------------------|--|
| Start the deck          | <code>&lt;deck&gt;</code>  |
| Set the canvas size     | <code>&lt;canvas width="1024" height="768" /&gt;</code>  |
| Begin a slide           | <code>&lt;slide bg="white" fg="black"&gt;</code>   |
| Place an image          | <code>&lt;image xp="70" yp="60" width="256" height="179" name="work.png" caption="Desk"/&gt;</code>  |
| Draw some text          | <code>&lt;text xp="20" yp="80" sp="3"&gt;Deck uses these elements&lt;/text&gt;</code>  |
| Make a bullet list      | <code>&lt;list xp="20" yp="70" sp="2" type="bullet"&gt;</code><br><code>&lt;li&gt;text, list, image&lt;/li&gt;</code><br><code>&lt;li&gt;line, rect, ellipse&lt;/li&gt;</code><br><code>&lt;li&gt;arc, curve&lt;/li&gt;</code> |
| End the list            | <code>&lt;/list&gt;</code>   |
| Draw a line             | <code>&lt;line xp1="20" yp1="10" xp2="30" yp2="10"/&gt;</code>   |
| Draw a rectangle        | <code>&lt;rect xp="35" yp="10" wp="4" hr="75" color="rgb(127,0,0)"/&gt;</code>   |
| Draw an ellipse         | <code>&lt;ellipse xp="45" yp="10" wp="4" hr="75" color="rgb(0,127,0)"/&gt;</code>  |
| Draw an arc             | <code>&lt;arc xp="55" yp="10" wp="4" hp="3" a1="0" a2="180" color="rgb(0,0,127)"/&gt;</code>   |
| Draw a quadratic bezier | <code>&lt;curve xp1="60" yp1="10" xp2="75" yp2="20" xp3="70" yp3="10" /&gt;</code>   |
| End the slide           | <code>&lt;/slide&gt;</code>  |
| End of the deck         | <code>&lt;/deck&gt;</code>   |

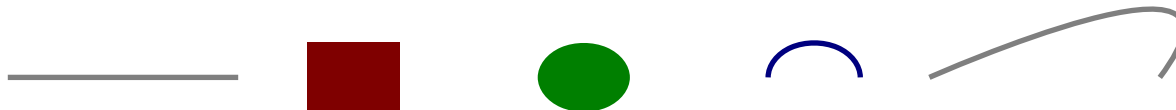
## Anatomy of a Deck

# Deck uses these elements

- text, list, image
- line, rect, ellipse
- arc, curve



Desk



# Text and List Markup

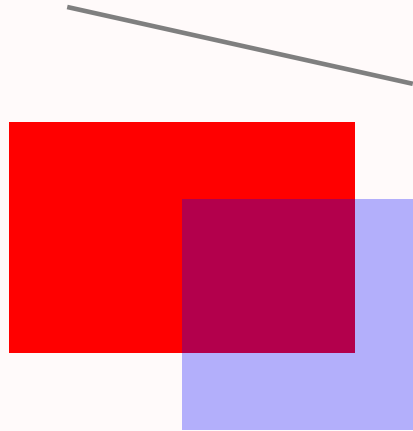
|                |  |
|----------------|--|
| Position, size | <code>&lt;text xp="..." yp="..." sp="..."&gt;</code>                           |
| Block of text  | <code>&lt;text ... type="block"&gt;</code>                                     |
| Lines of code  | <code>&lt;text ... type="code"&gt;</code>                                      |
| Attributes     | <code>&lt;text ... color="..." opacity="..." font="..." align="..."&gt;</code> |
| <hr/>          |  |
| Position, size | <code>&lt;list xp="..." yp="..." sp="..."&gt;</code>                           |
| Bullet list    | <code>&lt;list ... type="bullet"&gt;</code>                                    |
| Numbered list  | <code>&lt;list ... type="number"&gt;</code>                                    |
| Attributes     | <code>&lt;list ... color="..." opacity="..." font="..." align="..."&gt;</code> |

# Common Attributes for text and list

|                      |   |
|----------------------|---|
| <code>xp</code>      | horizontal percentage                             |
| <code>yp</code>      | vertical percentage                               |
| <code>sp</code>      | font size percentage                              |
| <code>type</code>    | "bullet", "number" (list), "block", "code" (text) |
| <code>align</code>   | "left", "middle", "end"                           |
| <code>color</code>   | SVG names ("maroon"), or RGB "rgb(127,0,0)"       |
| <code>opacity</code> | percent opacity (0-100, transparent - opaque)     |
| <code>font</code>    | "sans", "serif", "mono"                           |



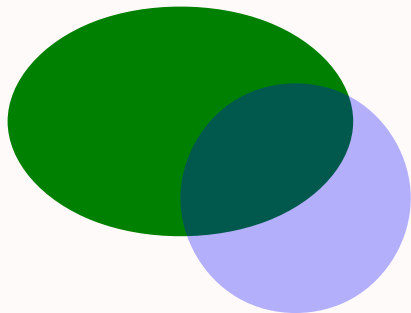
# Graphics Markup



```
<line xp1="5" yp1="75" xp2="20" yp2="70" sp="0.2"/>
```

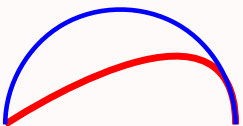
```
<rect xp="10" yp="60" wp="15" hr="66.6" color="red"/>
```

```
<rect xp="15" yp="55" wp="10" hr="100" color="blue" opacity="30"/>
```



```
<ellipse xp="10" yp="35" wp="15" hr="66.66" color="green"/>
```

```
<ellipse xp="15" yp="30" wp="10" hr="100" color="blue" opacity="30"/>
```



```
<curve xp1="5" yp1="10" xp2="15" yp2="20" xp3="15" yp3="10" sp="0.3" color="red"/>
```

```
<arc xp="22" yp="10" wp="10" hp="10" a1="0" a2="180" sp="0.2" color="blue"/>
```

A square grid with a light gray background and a thin blue border. The grid is composed of 10 columns and 10 rows. The horizontal axis (x-axis) is labeled with values 10, 20, 30, 40, 50, 60, 70, 80, and 90 at the top. The vertical axis (y-axis) is labeled with values 10, 20, 30, 40, 50, 60, 70, 80, and 90 on the left. The text "Percent Grid" is centered in the grid in a large, black, sans-serif font.

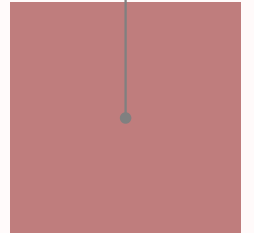
10%, 50%

Hello

50%, 50%



90%, 50%



Percentage-based layout

## bullet

- Point A
- Point B
- Point C
- Point D

## plain

First item

Second item

The third item

the last thing

## number

1. This
2. That
3. The other
4. One more

```
<list>...</list>
```

## bullet

- Point A
- Point B
- Point C
- Point D

## plain

First item

Second item

The third item

the last thing

## number

1. This
2. That
3. The other
4. One more

```
<list>...</list>
```

Design Examples



hello, world

Top

Left

Right

Bottom



20%

30%

70%

20%

Header (top 20%)

Summary  
(30%)

Detail  
(70%)

Footer (bottom 20%)

# Two Columns

One

Two

Three

Four



Tree and Sky

Five

Six

Seven

Eight



Rocks

This is not a notecard

Rich

Can't buy me love

Bliss

Worse

Better

Misery

We have each other

Poor

## Code

```
package main

import (
    "os"
    "github.com/ajstarks/svgo"
)

func main() {
    canvas := svg.New(os.Stdout)
    width := 850
    height := 1100
    canvas.Start(width, height)
    canvas.Gstyle("fill-opacity:0.3;font-size:480pt")
    for r := 0.0; r < 360.0; r += 45 {
        canvas.TranslateRotate(width/2, height/2, r)
        canvas.Text(0, 0, "a")
        canvas.Gend()
    }
    canvas.Gend()
    canvas.End()
}
```

## Output



A few months ago, I had a look at the brainchild of a few serious heavyweights working at Google. Their project, the Go programming language, is a static typed, c lookalike, semicolon-less, self formatting, package managed, object oriented, easily paralellizable, cluster fuck of genius with an unique class inheritance system. It doesn't have one.

# The Go Programming Language

is a static typed,  
c lookalike,  
semicolon-less,  
self formatting,  
package managed,  
object oriented,  
easily paralellizable,  
cluster fuck of genius  
with an unique class inheritance system.



# The Go Programming Language

is a static typed,  
c lookalike,  
semicolon-less,  
self formatting,  
package managed,  
object oriented,  
easily paralellizable,  
cluster fuck of genius  
with an unique class inheritance system.

---

# The Go Programming Language

is a static typed, c lookalike, semicolon-less, self formatting,  
package managed, object oriented, easily paralellizable,  
cluster fuck of genius with an unique class inheritance system.

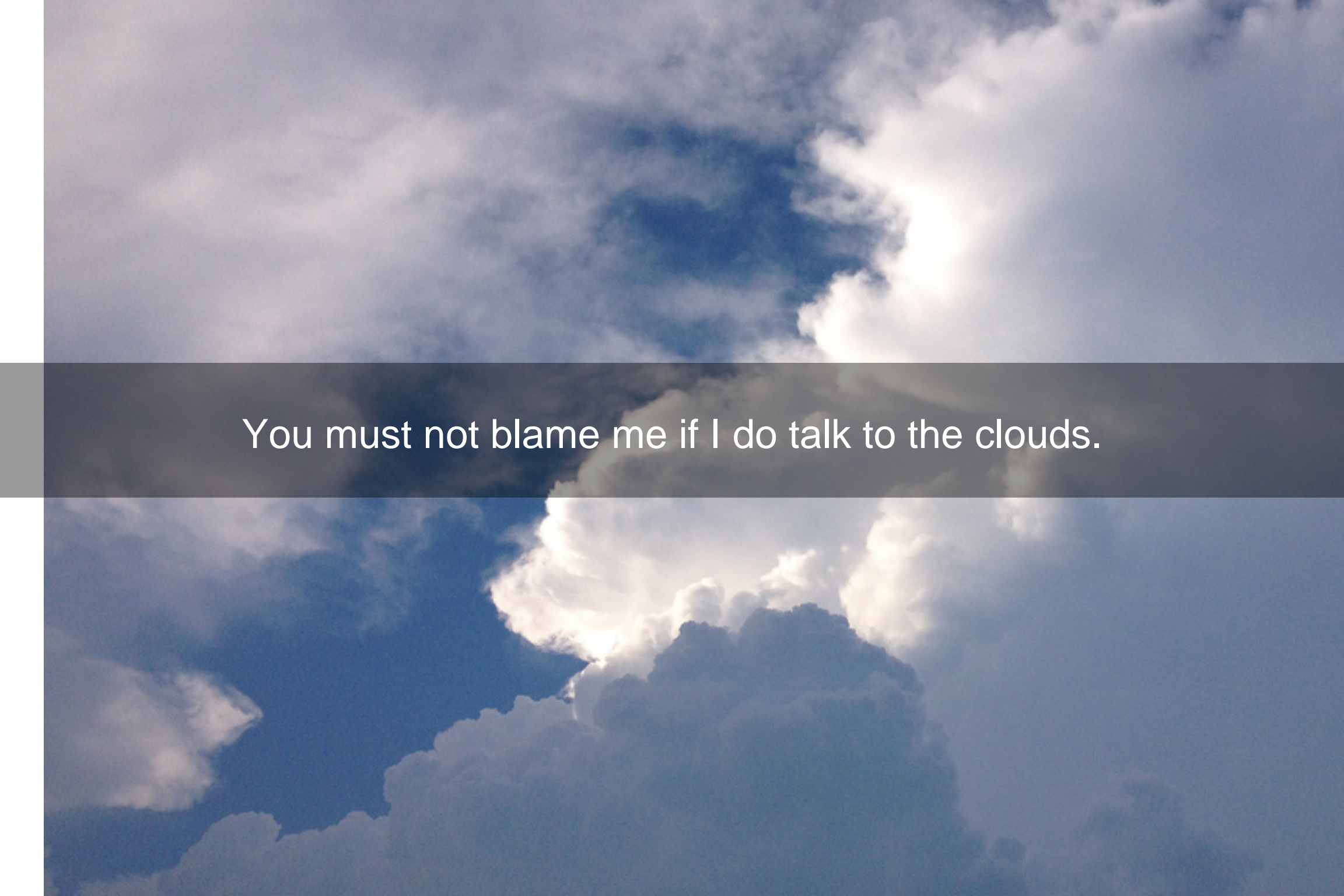
It doesn't have one.

So, the next time you're about to  
make a subclass, think hard and ask  
yourself

what would Go do

Andrew Mackenzie-Ross, <http://pocket.co/sSc56>





You must not blame me if I do talk to the clouds.

FOR, LO,

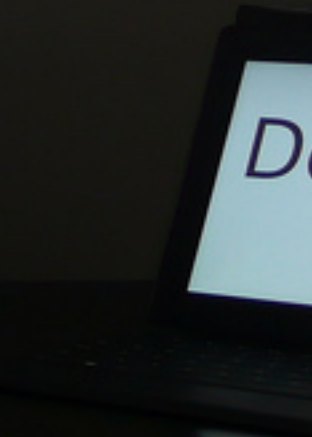
the winter is past,

the rain is over and gone;

The flowers appear on the earth;

the time for the singing of birds is come,

and the voice of the turtle is heard in our land.



Clients

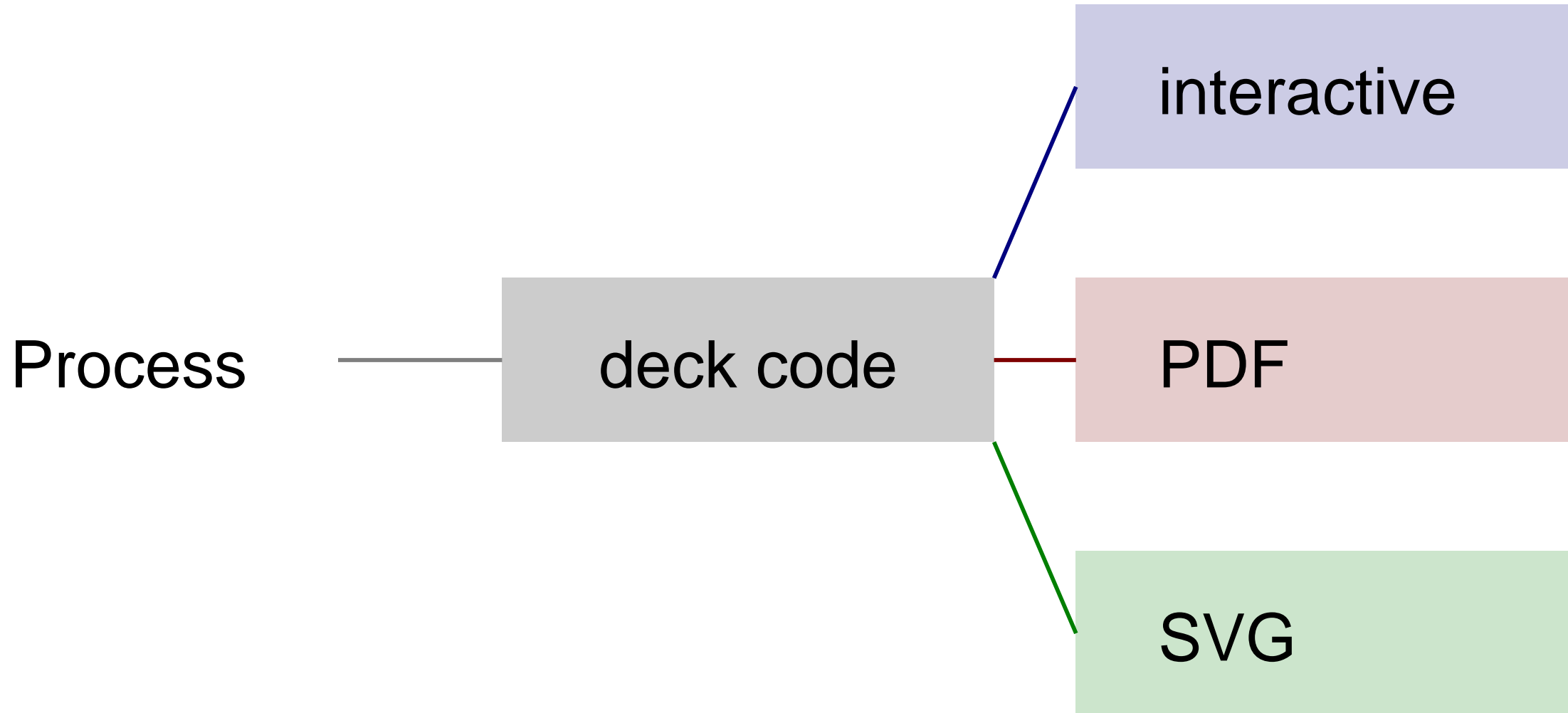
```
package main

import (
    "log"
    "github.com/ajstarks/deck"
)

func main() {
    presentation, err := deck.Read("deck.xml", 1024, 768) // open the deck
    if err != nil {
        log.Fatal(err)
    }
    for _, slide := range presentation.Slide {           // for every slide...
        for _, t := range slide.Text {                   // process the text elements
            x, y, size := deck.Dimen(presentation.Canvas, t.Xp, t.Yp, t.Sp)
            slideText(x, y, size, t)
        }
        for _, l := range slide.List {                   // process the list elements
            x, y, size := deck.Dimen(presentation.Canvas, l.Xp, l.Yp, l.Sp)
            slideList(x, y, size, l)
        }
    }
}
```

## A Deck Client







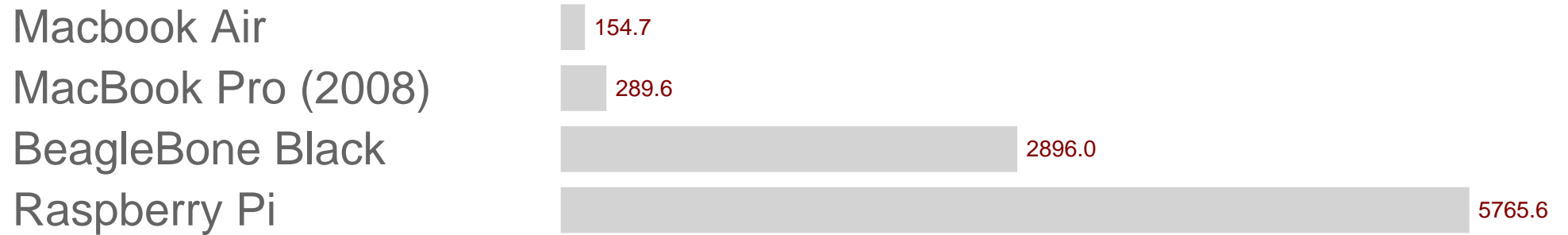
```

func main() {
    benchmarks := []Bardata{
        {"Macbook Air", 154.701},
        {"MacBook Pro (2008)", 289.603},
        {"BeagleBone Black", 2896.037},
        {"Raspberry Pi", 5765.568},
    }
    ts := 2.5
    hts := ts / 2
    x := 10.0
    bx1 := x + (ts * 12)
    bx2 := bx1 + 50.0
    y := 60.0
    maxdata := 5800.0
    linespacing := ts * 2.0
    text(x, y+20, "Go 1.1.2 Build and Test Times", ts*2, "black")
    for _, data := range benchmarks {
        text(x, y, data.label, ts, "rgb(100,100,100)")
        bv := vmap(data.value, 0, maxdata, bx1, bx2)
        line(bx1, y+hts, bv, y+hts, ts, "lightgray")
        text(bv+0.5, y+(hts/2), fmt.Sprintf("%.1f", data.value), hts, "rgb(127,0,0)")
        y -= linespacing
    }
}

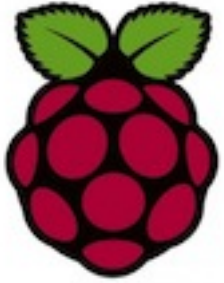
```

## Generating a Barchart

# Go 1.1.2 Build and Test Times



```
$ (echo '<deck><slide>'; go run deckbc.go; echo '</slide></deck>')
```



```
go get github.com/ajstarks/deck/vgdeck
```



```
go get github.com/ajstarks/deck/pdfdeck
```



```
go get github.com/ajstarks/deck/svgdeck
```

`pdfdeck [options] file.xml...`

- sans, -serif, -mono [font] specify fonts
- pagesize [w,h, or Letter, Legal, Tabloid, A2-A5, ArchA, Index, 4R, Widescreen]
- stdout (output to standard out)
- outdir [directory] directory for PDF output
- fontdir [directory] directory containing font information
- author [author name] set the document author
- title [title text] set the document title
- grid [percent] draw a percent grid on each slide

`svgdeck [options] file.xml...`

- `-sans, -serif, -mono [font]` specify fonts
- `-pagesize [Letter, Legal, A3, A4, A5]`
- `-pagewidth [canvas width]`
- `-pageheight [canvas height]`
- `-stdout` (output to standard out)
- `-outdir [directory]` directory for PDF output
- `-title [title text]` set the document title
- `-grid [percent]` draw a percent grid on each slide

`vgdeck [options] file.xml...`

- loop [duration] loop, pausing [duration] between slides
- slide [number] start at slide number
- w [width] canvas width
- h [height] canvas height
- g [percent] draw a percent grid

# vgdeck Commands

`+, Ctrl-N, [Return]`

Next slide

`-, Ctrl-P, [Backspace]`

Previous slide

`^, Ctrl-A`

First slide

`$, Ctrl-E`

Last slide

`r, Ctrl-R`

Reload

`x, Ctrl-X`

X-Ray

`/, Ctrl-F [text]`

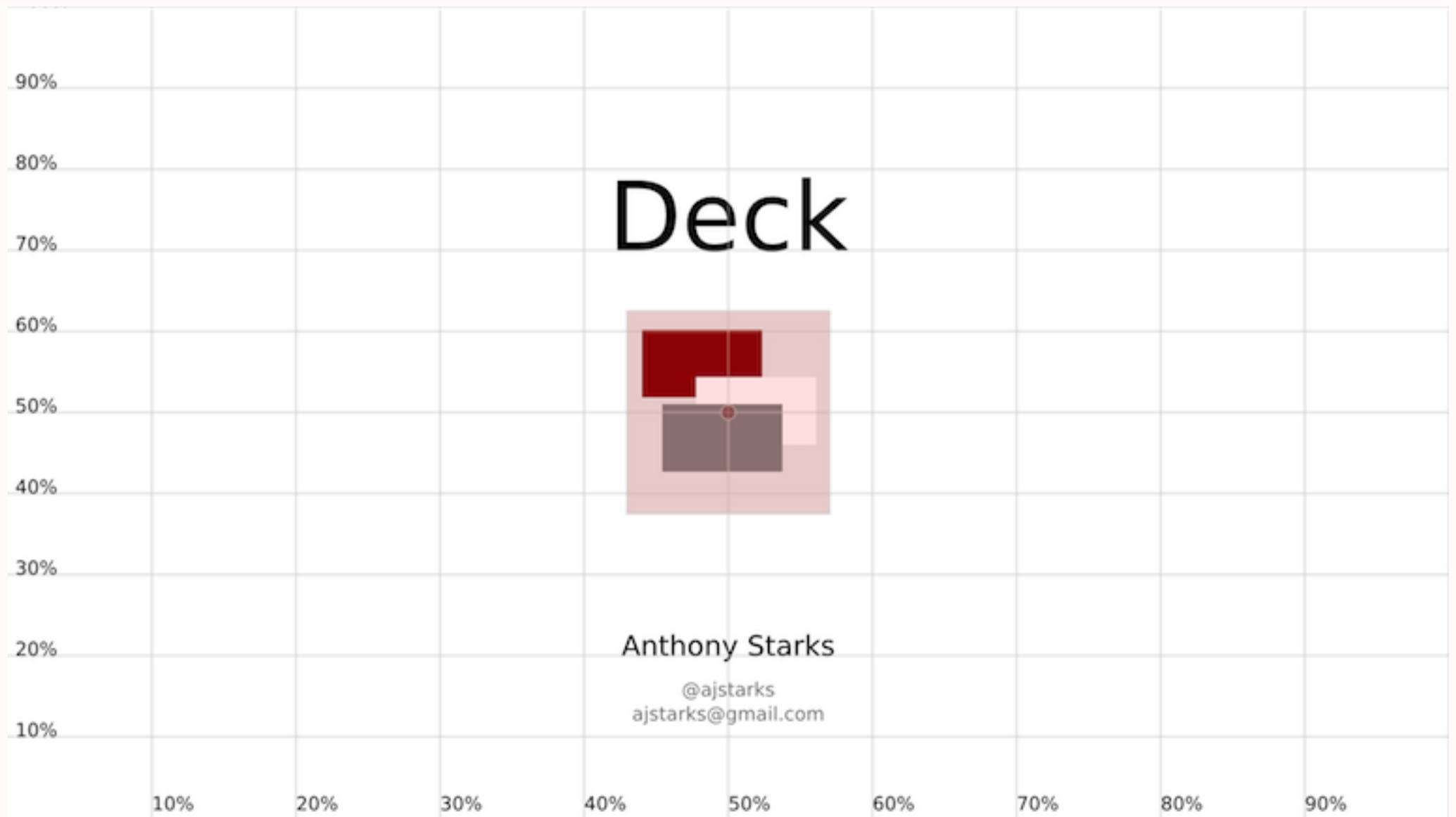
Search

`s, Ctrl-S`

Save

`q`

Quit



X-Ray mode shows the percent grid, and highlights images



[github.com/ajstarks/deck](https://github.com/ajstarks/deck)



[ajstarks@gmail.com](mailto:ajstarks@gmail.com)  
@ajstarks