Halfa Document

Text Based Presentation

Open Source

thevpc
https://github.com/thevpc/halfa

2024-06-15
v1.0

Rationale

- Text-based, declarative, and intuitive syntax
- Readable by humans, writable with ease
- Designed for long-lived documents with effortless maintenance
- Unmatched control over rendering
- Seamless multi-file support
- Version-control friendly (Git & more)
- Integrates with LaTeX, UML, and beyond

Rationale

- Uses TSON which is a derivative of JSON format but with more readability
- Declarative syntax : what you write is what you get
- Parameterizable : variables and conditions are processed for rendering
- Templatable : one can define his own components
- Themable : uses a CSS like styling
- Composable : you can combine multiple components to build reusable components

Hello World

Hello World

```
"Hello World"
```

- HelloWorld
- HelloWorld
- HelloWorld

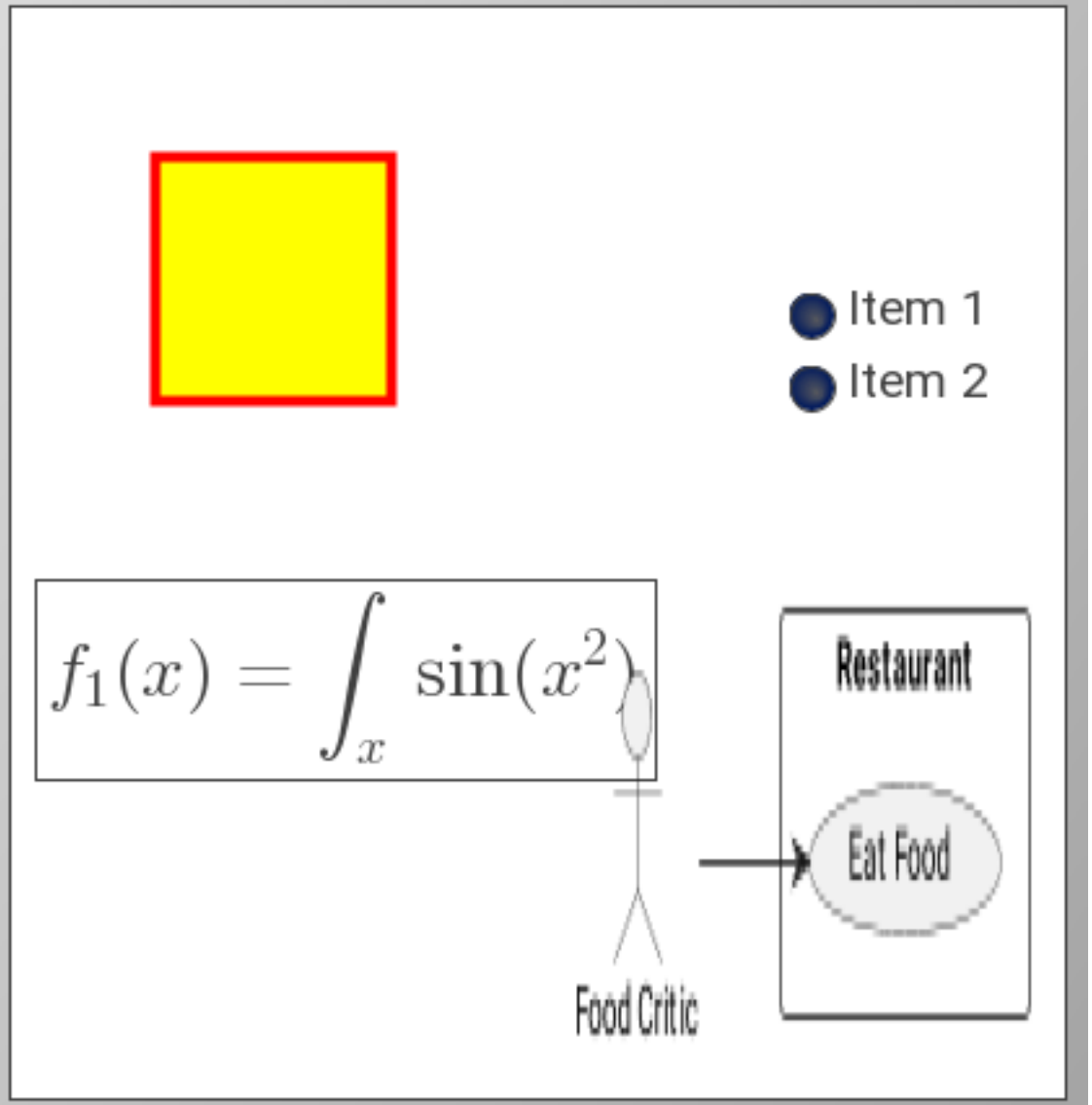
```
"
    - Hello ##World##
    - Hello ###World###
    - Hello ####World##
"
```

- HelloWorld
- HelloWorld
- HelloWorld

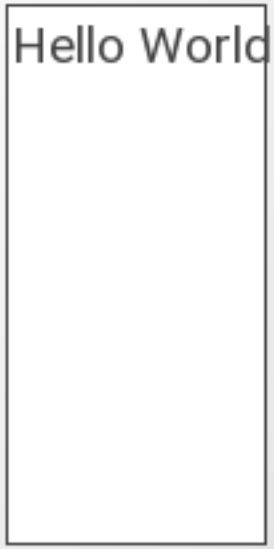
```
ul {
    "Hello ##World##"
    "Hello ###World###"
    "Hello ####World####"
}
```

More Elaborate Example

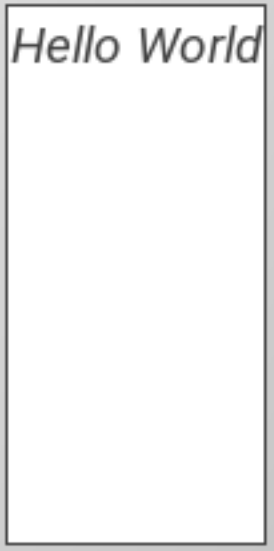
```
// this is a comment
page{
    grid ((2,2)){
        rectangle  (at :center ,size :(50),
            background :yellow  ,
            color :red , stroke  :4
        )
        ul (at :center ){ "Item 1" , "Item 2" }
        eq("x^2_1=\sin(x)") // supports LATEX
        uml( // supports PLANTUML
            """
                left to right direction
                actor "Food Critic" as fc
                rectangle Restaurant {
                  usecase "Eat Food" as UC1
                }
                fc --> UC1
            """
        )
    }
}
```



Simple Text

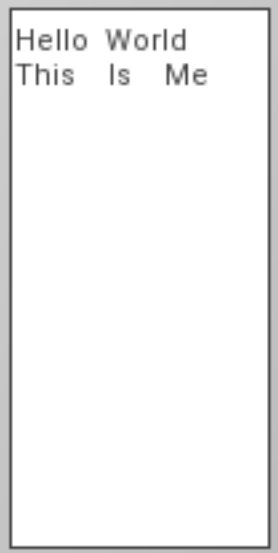



```
plain  ("Hello World")
```




```
plain  ("Hello World",font-italic    )
```


```
plain  (
         "
             Hello
             World
         "
       ,font-bold  )
```




```
plain  (
         "
             Hello
             World
         "
       font-family   : monospaced
       font-size     : 5%g
       )
```

Rich Text

Hello World

Hello World

Hello World

Hello World

```
"Hello World"
text ("Hello World")
text ("Hello World",font-italic    )
text ("Hello World",font-bold   )
```

Bold X
Italic Y
Title 1
Title 2
Title 3

```
**Bold   X**
__Italic     Y__
#Title    1#
##Title    2##
###Title    3###
```

color 1 color 2
color 3 color 4
italic hint

```
text (
ntf
"""
    ##:p1:color 1## ##:p2:color 2#
    ##:p3:color 3## ##:p10:color 4
    ##:/:italic## ##:info hint##
"""
)
```

Equation 1 =
$X^2 = \sin(x)$

Equation 2 =
$X^2 = \sin(x)$

Equation 3 =
$X^2 = \sin(x)$

```
#Equation    1# =
$X^2=\sin (x) $
##Equation    2## =
$X^2=\sin (x) $
###Equation    3### =
$X^2=\sin (x) $
```

Latex Equations


```
"$X^2=\sin(x)$"
```


```
eq("X^2=\sin(x)")
```

Source Code

```
source (
java
"""
    public static class
    MyClass{
        int value = 10;
        int add(int b){
            value++;
        }
    }
"""
)
```

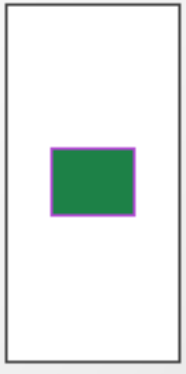
```
source (
xml
"""
    <a value="text">
      <b value="text"></b>
    </a>
"""
)
```

```
source (
sql
"""
    Select *
    From Tab
    Where 1=1
"""
)
```

text, java,
c#, c++
xml,html,json
bash,fish,cmd
sql,
hd, ntf,hadra,
tson

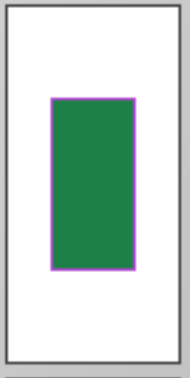
Supported Languages

Shapes



rectangle ()

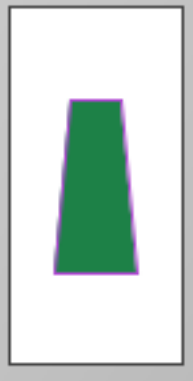
triangle ()



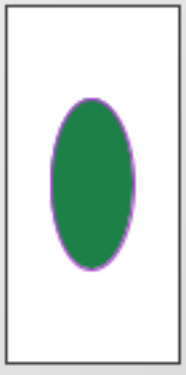
square ()

rhombus ()
diamond ()

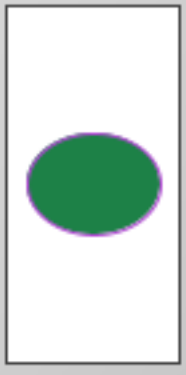
parallelogram ()



trapezoid ()

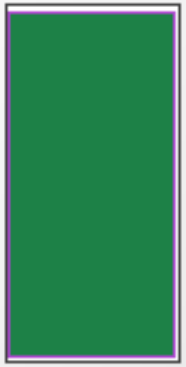


circle ()

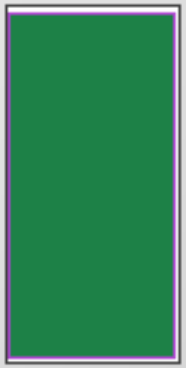


ellipse (size :(80,30))

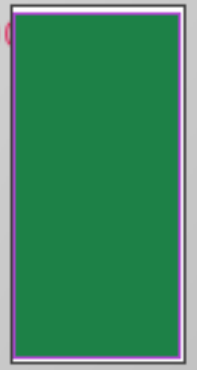
Sizes



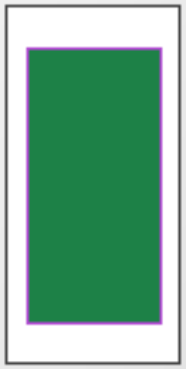
```
rectangle ()
```



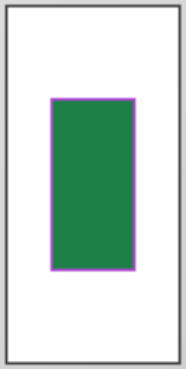
```
rectangle (size :(100,100
```



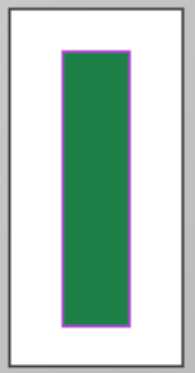
```
rectangle (size :100)
```



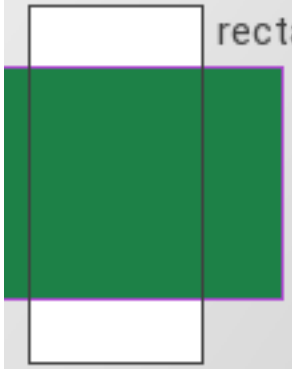
```
rectangle (size :80)
```



```
rectangle (size :50)
```

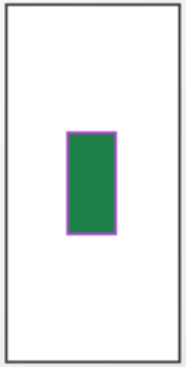


```
rectangle (size :(40,80))
```

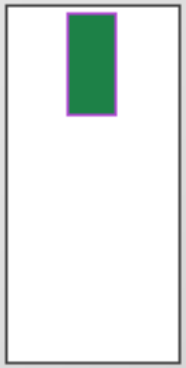


```
rectangle (size :15%g)
```

Positions

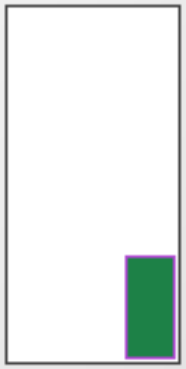


```
rectangle (at :center )
```

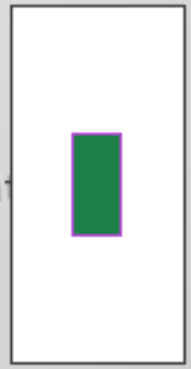


```
rectangle (at :top )
```

```
rectangle (at :top-left )
```

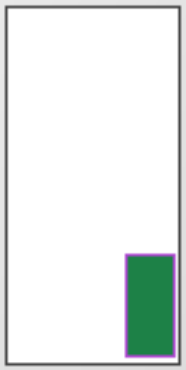


```
rectangle (at :bottom-right
```

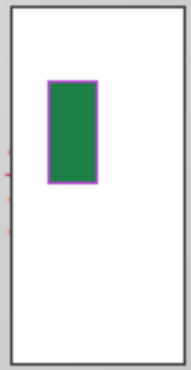


```
rectangle (origin :(50,5
           position :(50,50))
```

```
rectangle (origin :(60,60
           position :(100,100))
```

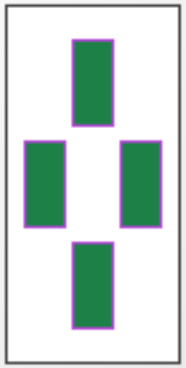


```
rectangle (origin :(100,
           position :(100,100)
```

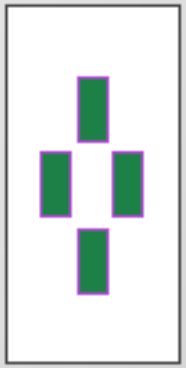


```
rectangle (origin :(100,100)
           position :(50,50))
```

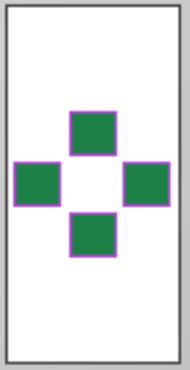
Margins



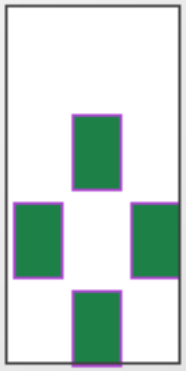
rectangle (margin :10)



rectangle (margin :20)

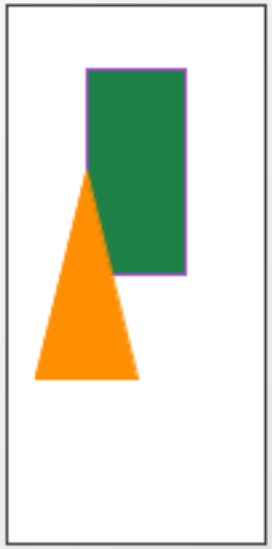


rectangle (margin :(5,30)

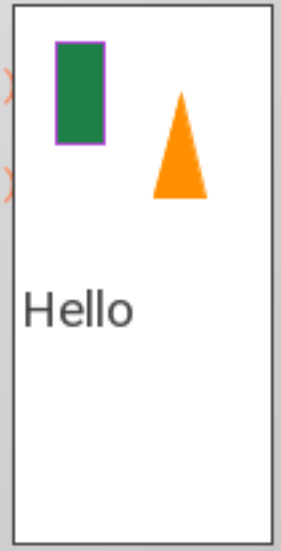


rectangle (margin :(5,30,0,0))

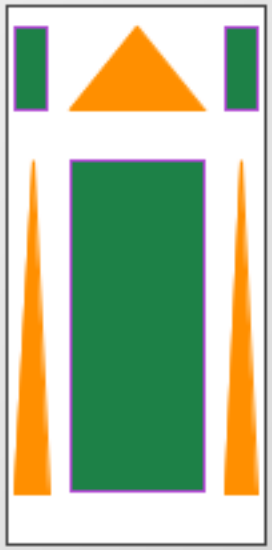
Layout



```
stack {
    rectangle   (at :top , size :(50,50)
        background  :$palette  [4])
    triangle    (at :left , size :(50,50)
        background  :$palette  [5])
}
```

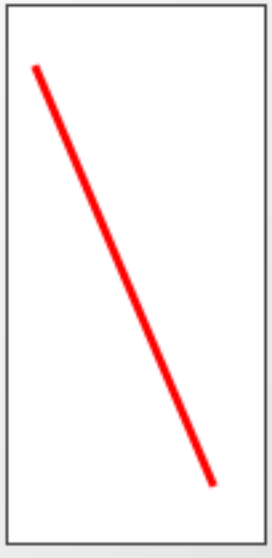



```
grid  ((2,2)){
    rectangle   (at :top , size :(50,50)
        background  :$palette  [4])
    triangle    (at :left , size :(50,50)
        background  :$palette  [5])
    "Hello"
}
```



```
grid  ((3,2),
    columns-weight   :[1,4],
    rows-weight  :[1,4]){
        rectangle   ()
        triangle    ()
        rectangle   ()
        triangle    ()
        rectangle   ()
        triangle    ()
}
```

Lines



```
line  (from :(0,0),  to :(80,20))
```

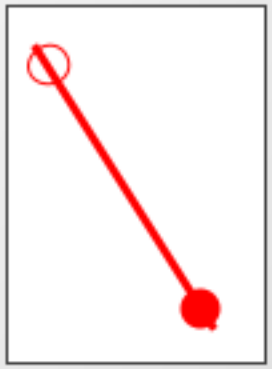
```
arc (from :30,  to :180)
```

```
quad-curve  (
      from :(10,10),
      ctrl  :(60,30)
      to :(80,90),
)
```

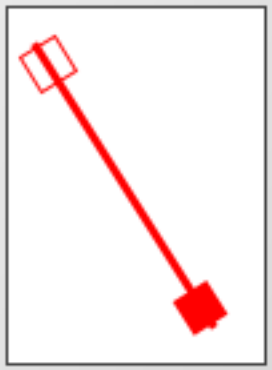
```
cubic-curve   (
from :(10,10),
ctrl  1:(60,30),
ctrl  2:(30,60)
to :(80,90),
)
```

Arrows

```
line (
    from :(10,10), to :(80,90)
    end-arrow :simple ()
)
```

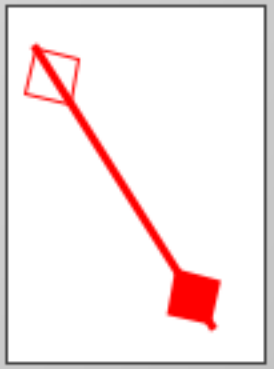


```
line (
    from :(10,10), to :(80,90)
    start-arrow :circle ()
    end-arrow :circle-full ()
)
```



```
line (
    from :(10,10), to :(80,90)
    start-arrow :rectangle ()
    end-arrow :rectangle-full ()
)
```

```
line (
    from :(10,10), to :(80,90)
    start-arrow :triangle ()
    end-arrow :triangle-full ()
)
```



```
line (
    from :(10,10), to :(80,90)
    start-arrow :diamond ()
    end-arrow :diamond-full ()
)
```

Polygons

pentagon ()

hexagon ()

heptagon ()

octagon ()

nonagon ()

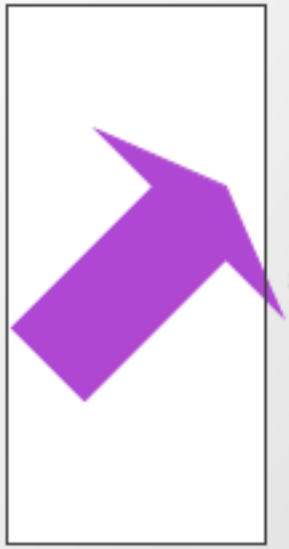
decagon ()

polygon ()

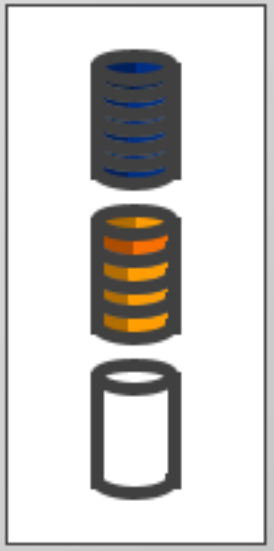
polygon (count :8)

```
polygon (points :[
(0,0),(50,0),
(100,80),(50,50)
])
```

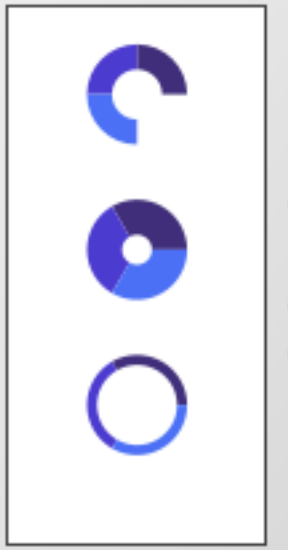
Other Shapes



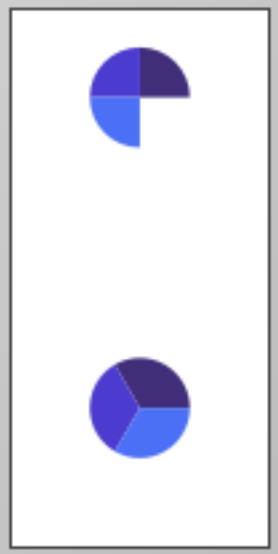
```
arrow (at : center ,rotate :-45)
```



```
cylinder (ellipse-height :20,
  segment-count :5)
cylinder (ellipse-height :20,
  segment-count :3)
cylinder (ellipse-height :20)
```



```
donut (inner-radius :50,
  start-angle :0,extent-angle :270)
donut (inner-radius :30)
donut (inner-radius :80)
```



```
pie ()
pie (start-angle :0,extent-angle :270
```

Images

```
image ("../../../images/image.png")
```

```
image ("../../../images/image.jpg")
```

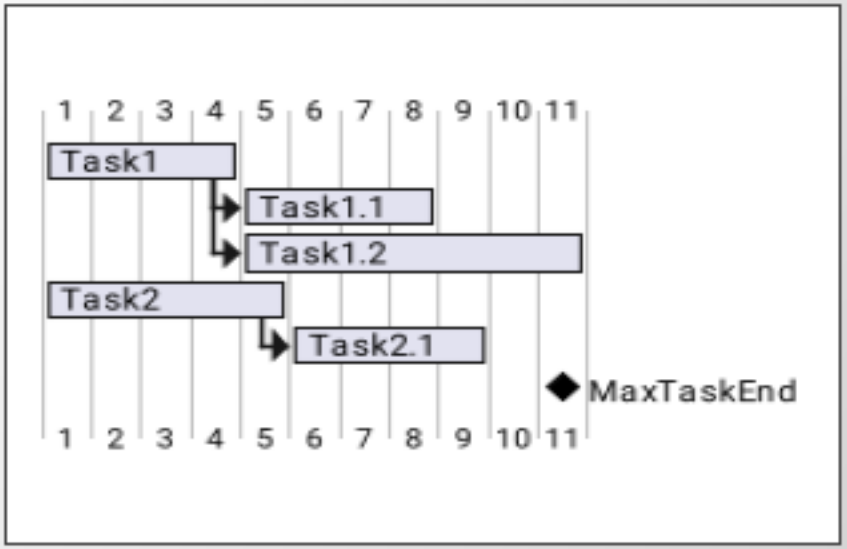
```
image ("../../../images/image.gif")
```

```
image ("../../../images/image.svg")
```

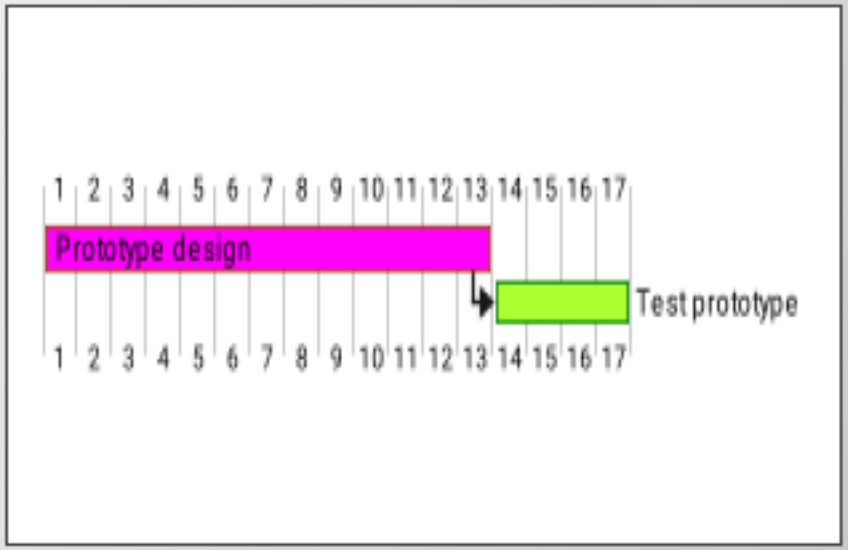
```
image ("../../../images/image.avif")
```

```
image ("../../../images/image.webp")
```

Gantt Diagrams

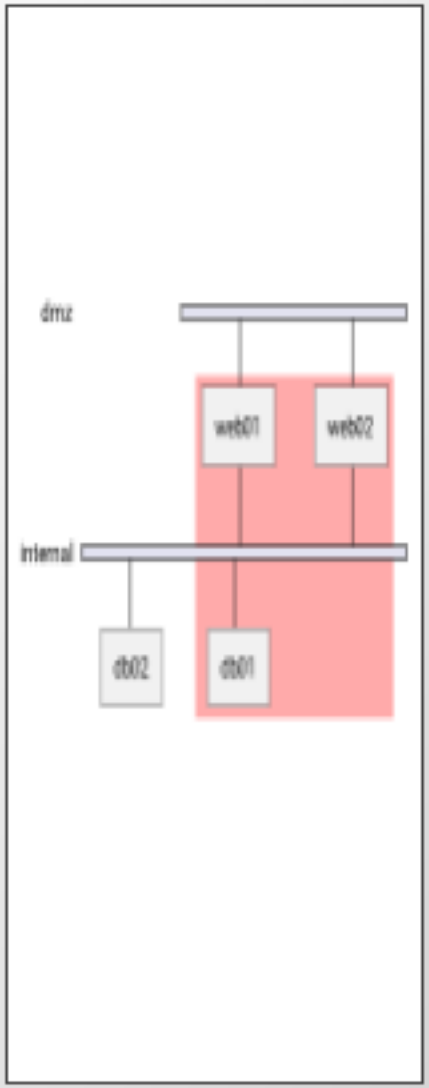



```
gantt (
    """
        [Task1] requires 4 days
        then [Task1.1] requires 4 days
        [Task1.2] starts at [Task1]'s end and requires
        [Task2] requires 5 days
        then [Task2.1] requires 4 days
        [MaxTaskEnd] happens at [Task1.1]'s end
        [MaxTaskEnd] happens at [Task1.2]'s end
        [MaxTaskEnd] happens at [Task2.1]'s end
    """
)
```

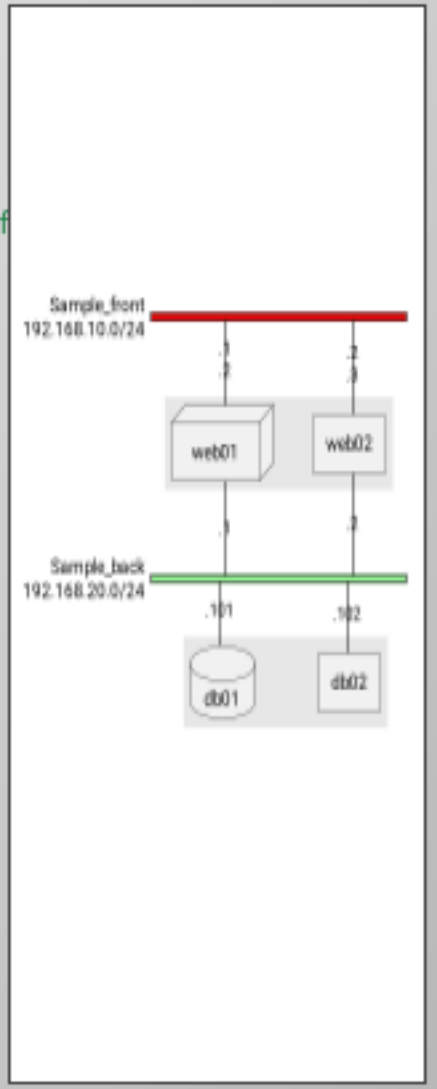



```
gantt (
    """
        [Prototype design] requires 13 days
        [Test prototype] requires 4 days
        [Test prototype] starts at [Prototype design]'s
        [Prototype design] is colored in Fuchsia/FireBr
        [Test prototype] is colored in GreenYellow/Gree
    """
)
```

Network Diagrams



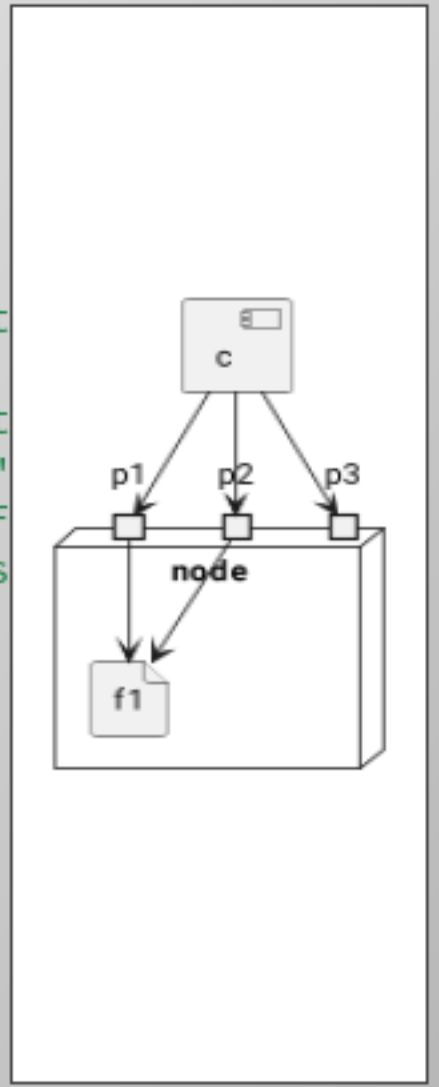

```
nwdiag {
    """
        // define group outside of
        group {
          color = "#FFAAAA";
          web01;
          web02;
          db01;
        }
        network dmz {
          web01;
          web02;
        }
        network internal {
          web01;
          web02;
          db01;
          db02;
        }
    """
}
```




```
nwdiag {
    """
        network Sample_front {
            address = "192.168.10.0/
            color = "red"
            // define group
            group web {
              web01 [address = ".1,
              web02 [address = ".2,
            }
          }
          network Sample_back {
            address = "192.168.20.0/
            color = "palegreen"
            web01 [address = ".1"]
            web02 [address = ".2"]
            db01 [address = ".101",
            db02 [address = ".102"]
            // define network using
            group db {
              db01;
              db02;
            }
          }
    """
}
```

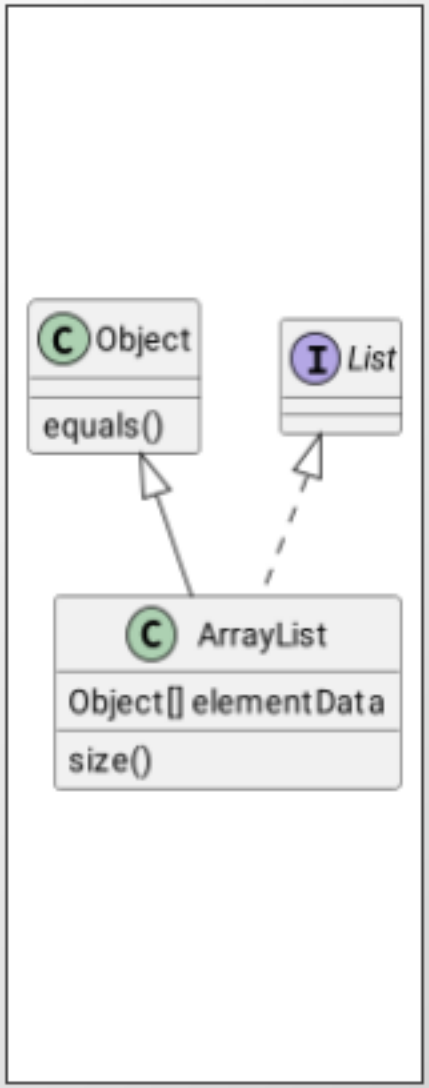
UML Use Case


```
uml (
    """
       left to right direct
       actor "Food Critic"
       rectangle Restaurant
         usecase "Eat Food"
         usecase "Pay for F
         usecase "Drink" as
       }
       fc --> UC1
       fc --> UC2
       fc --> UC3
    """
)
```

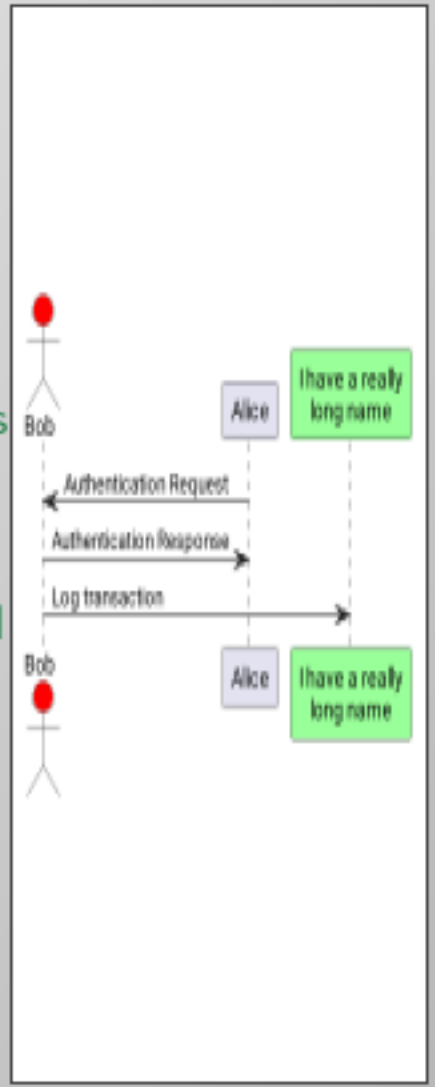



```
uml (
    """
       [c]
       node node {
         port p1
         port p2
         port p3
         file f1
       }
       c --> p1
       c --> p2
       c --> p3
       p1 --> f1
       p2 --> f1
    """
)
```

UML Classes

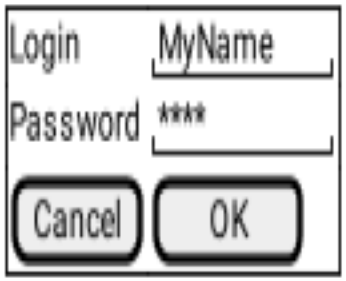



```
uml (
    """
        Object <|-- ArrayLis
        List <|.. ArrayList
        interface List
        Object : equals()
        ArrayList : Object[]
        ArrayList : size()
    """
)
```

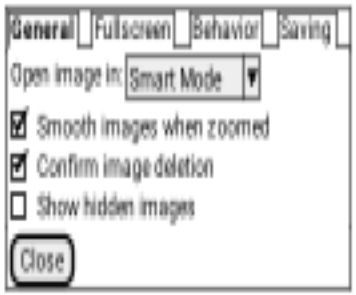



```
uml (
    """
        actor Bob #red
        ' The only difference
        'and participant is t
        participant Alice
        participant "I have a
        /' You can also decla
           participant L as '
          '/
        Alice->Bob: Authentic
        Bob->Alice: Authentic
        Bob->L: Log transacti
    """
)
```

Wireframe Diagrams




```
wireframe (
    """
        Login    | "MyName
        Password | "****
        [Cancel] | [  OK   ]
    """
)
```




```
wireframe (
    """
        {+
        {/ <b>General | Fulls
        {
        { Open image in: | ^S
        [X] Smooth images whe
        [X] Confirm image del
        [ ] Show hidden image
        }
        [Close]
        }
    """
)
```

Test me

open0110-test.hd file and try to write some things there

Thank you!

taha.bensalah@gmail.com