

ntexup

*A declarative, text-based document
& presentation generator*

Open Source

thevpc

<https://github.com/thevpc/ntexup>

2025-08-06

v0.8.6.0



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
- Templatatable : one can define his own components
- Themable : uses a CSS like styling
- Composable : you can combine multiple components to build reusable parts
- Portable: works across platforms and environments with minimal setup



Hello World



Hello World

"Hello World"

- Hello World
- Hello World
- Hello World

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

- Hello World

- Hello World

- Hello World

```
ul{  
    ¶ Hello ##World##  
    ¶ Hello ###World###  
    ¶ Hello #####World#####  
}
```

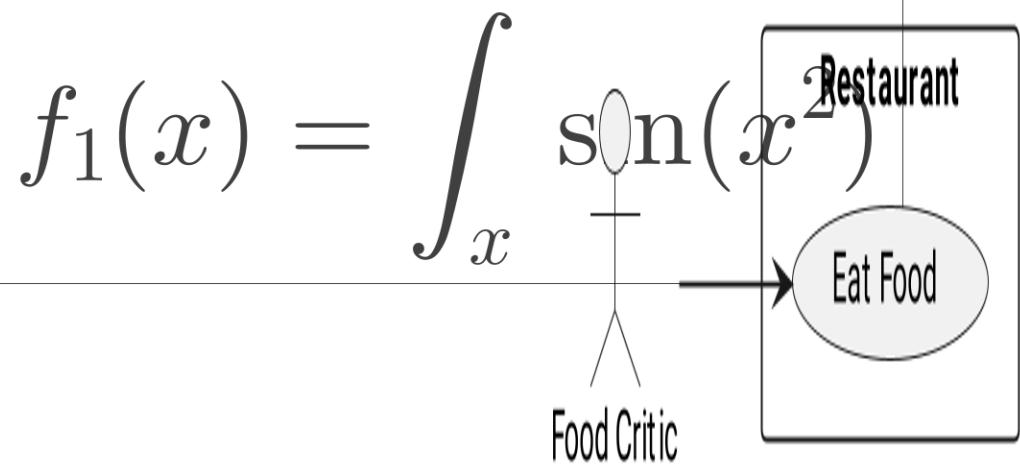
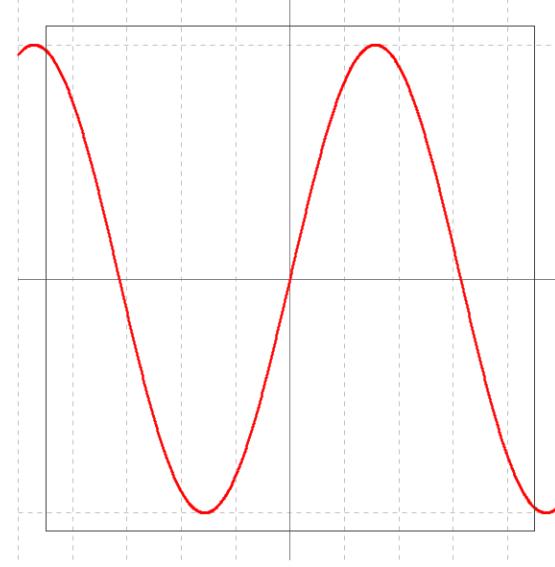


More Elaborate Example



```
// this is a comment
page{
    grid(2,2){
        group{ // supports shapes
            rectangle(at:center,size:(90),
                background:yellow,
                color:red, stroke:4
            )
            ul(origin:left, position:(20,50)){
                "Item 1" "Item 2"
            }
        }
        plot2d(xmin:-5 , xmax: 5){
            curve{ // supports plot
                f(x):sin(x)
                title:'sin' , color:red
            }
        }
        // supports LATEX
        eq("f_1(x)=\int_x \sin(x^2)",font-size:5%P)
        uml// supports PLANTUML
        """
            left to right direction
            actor "Food Critic" as fc
            rectangle Restaurant {
                usecase "Eat Food" as UC1
            }
            fc --> UC1
        """
    }
}
```

- Item 1
- Item 2





Simple Text



Hello World

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

Hello World

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

Hello
World

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

Hello World
This Is Me

```
plain(  
    "  
        Hello  
        World  
    ",  
    font-family: monospaced  
)
```



NTF format



Hello World

```
ntf(  
"""  
##Hello## ### World ###  
"""  
)
```

color 1 color .
color 3 color
italic hint

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



Latex Equations

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

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

Latex math expressions are supported out of the box...

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

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

you can use superscripts and subscripts and custom symbols...

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

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

you can embed eq in a string...

Equation 1 (su

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

Equation 2 =

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

Equation 3 =

$X^2 = \sin(x)$

""")

```
##Equation 1 (superscript)## =
[[eq: X^2=\sin(\pi x) ]]
###Equation 2### =
[[eq: X^2=\sin \left( \pi x \right) ]]
####Equation 3#### =
[[eq: X^2=\sin(x) ]]
```

you can embed eq in a string...



Source Code

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

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

```
Select *  
From Tab  
Where 1=1
```

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

```
<a value="text">  
  <b value="text"></b>  
</a>
```

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

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

Supported Languages



Rich Text

Hello World

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###  
"""
```

```
"""  
[[ntf: ##:p1: hello##]] world  
"""
```

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

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

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

```
"""  
#Equation 1# =  
[[eq: X^2=\sin(\pi x) ]]  
##Equation 2## =  
[[eq: X^2=\sin \left( \pi x \right) ]]  
##Equation 3## =  
[[eq: X^2=\sin(x) ]]  
"""
```



Bullets



• Hello World

• Hello World

• Hello World

I. Hello World

1. Sub 1

2. Sub 2

II. Hello World

III. Hello World

```
ul{  
    ¶ Hello ##World##  
    ¶ Hello ###World###  
    ¶ Hello #####World#####  
}
```

- Hello World
- Sub 1
- Sub 2
- Hello World
- Hello World

```
ul{  
    ¶ Hello ##World##  
    ul{  
        ¶ Sub 1  
        ¶ Sub 2  
    }  
    ¶ Hello ###World###  
    ¶ Hello #####World#####  
}
```

```
ol{  
    ¶ Hello ##World##  
    ol{  
        ¶ Sub 1  
        ¶ Sub 2  
    }  
    ¶ Hello ###World###  
    ¶ Hello #####World#####  
}
```



Shapes



rectangle()



triangle()



square()



rhombus()
diamond()



parallelogram()



trapezoid()



circle()



ellipse(size:(80,30))



Sizes

`rectangle()`

`rectangle(size:80)`

`rectangle(size:15%P)`

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

`rectangle(size:50)`

`rectangle(size:100)`

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



Positions



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

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

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

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

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

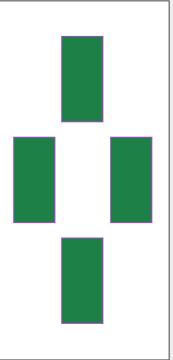
```
rectangle(origin:(60,60)  
position:(100,100))
```

```
rectangle(origin:(100,10)  
position:(100,100))
```

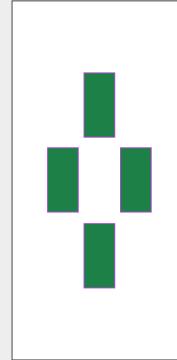
```
rectangle(origin:(100,100)  
position:(50,50))
```



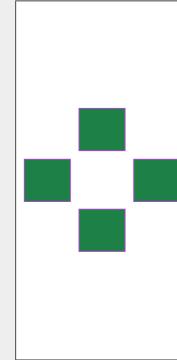
Margins



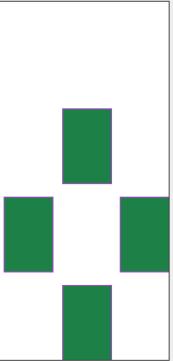
```
rectangle(margin:10)
```



```
rectangle(margin:20)
```



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



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



Layout

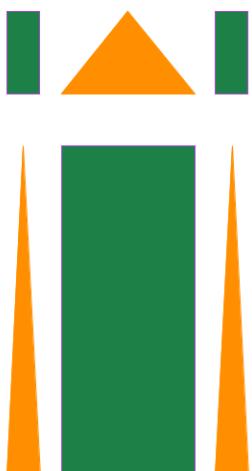


```
group{
    rectangle(at:top, size:(50,50),
              background:themeColors[4])
    triangle(at:left, size:(50,50),
             background:themeColors[5])
}
```



Hello

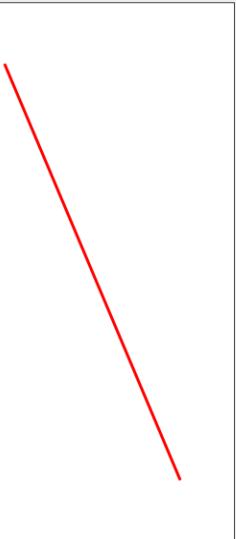
```
grid((2,2){
    rectangle(at:top, size:(50,50),
              background:themeColors[4])
    triangle(at:left, size:(50,50),
             background:themeColors[5])
    "Hello"
}
```



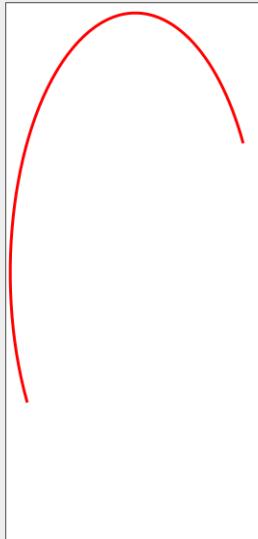
```
grid((3,2),
      columns-weight:[1,4],
      rows-weight:[1,4]){
    rectangle()
    triangle()
    rectangle()
    triangle()
    rectangle()
    triangle()
}
```



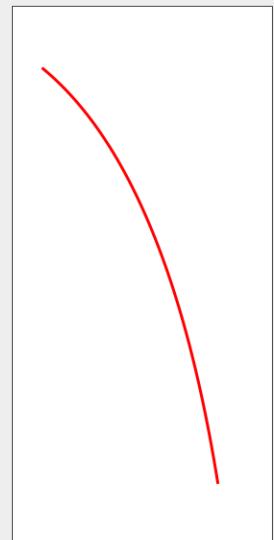
Lines



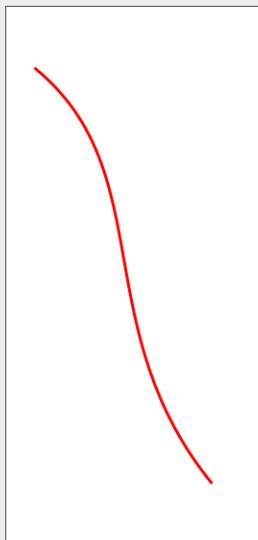
```
line(from:(0,0), to:(80,20))
```



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



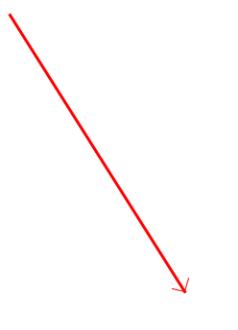
```
quad-curve(  
    from:(10,10),  
    ctrl:(60,30)  
    to:(80,90),  
)
```



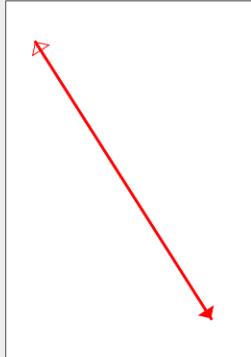
```
cubic-curve(  
    from:(10,10),  
    ctrl1:(60,30),  
    ctrl2:(30,60)  
    to:(80,90),  
)
```



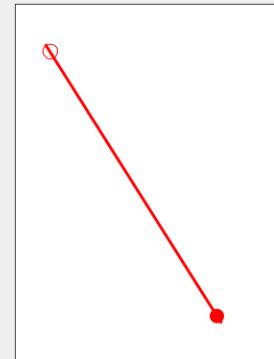
Arrows



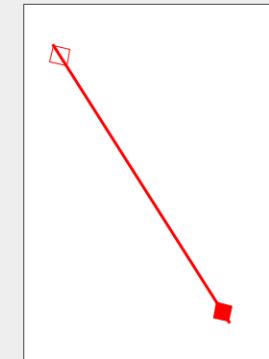
```
line(  
  from:(10,10), to:(80,90)  
  end-arrow:simple()  
)
```



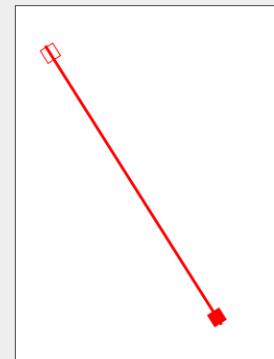
```
line(  
  from:(10,10), to:(80,90)  
  start-arrow:triangle()  
  end-arrow:triangle-full()  
)
```



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



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



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



Polygons



pentagon()



hexagon()



heptagon()



octagon()



nonagon()



decagon()



polygon()



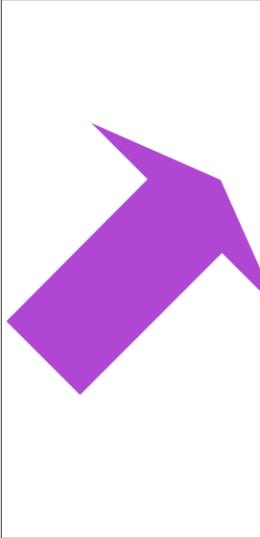
polygon(count:8)



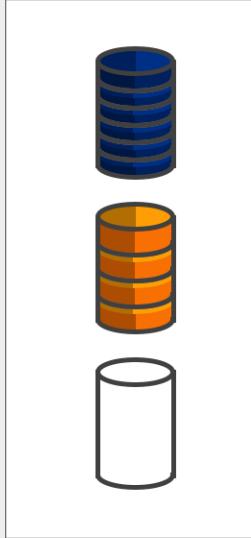
```
polygon(points:[  
  (0,0),(50,0),  
  (100,80),(50,50)  
])
```



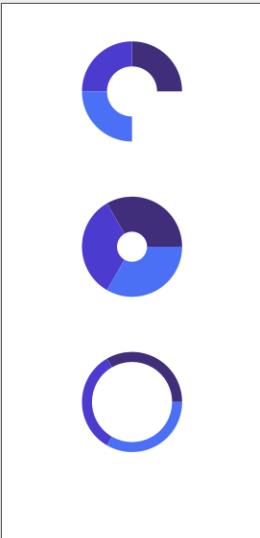
Other Shapes



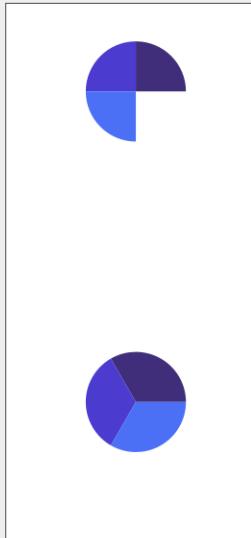
```
arrow(at: center,rotate:-45)
```



```
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.gif")
```



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

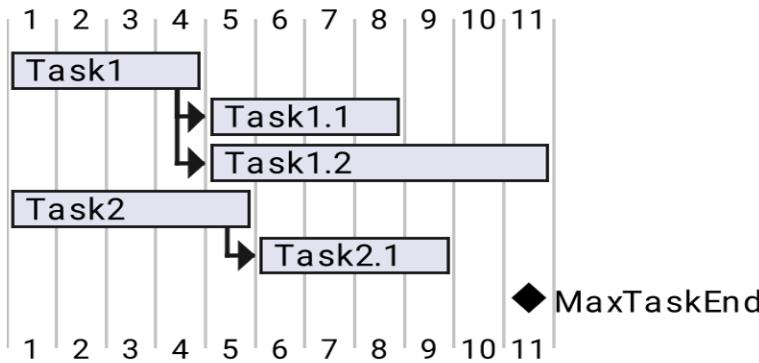
```
image("../.../.../images/image.jpg")
```

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

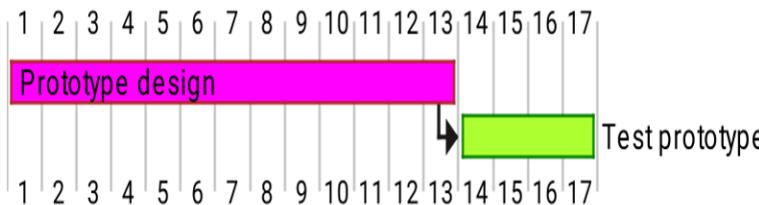
```
image("../.../.../images/image.webp")
```



Gantt Diagrams



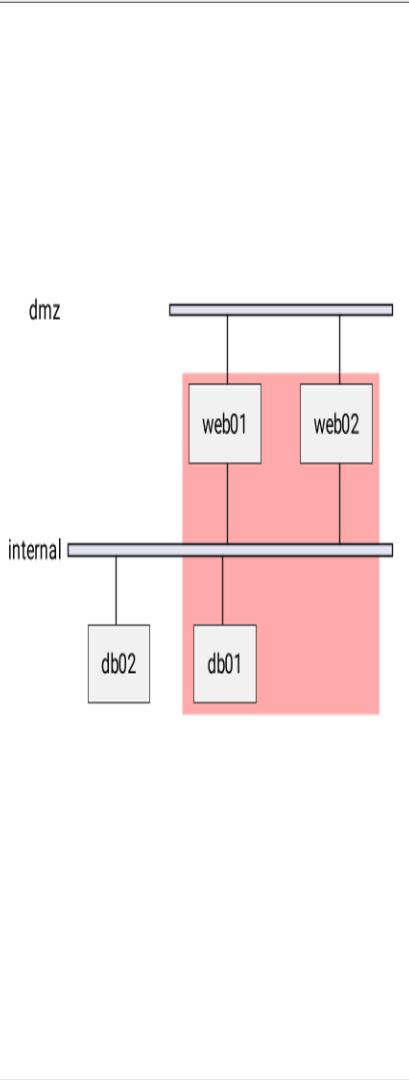
```
gantt()
"""
[Task1] requires 4 days
then [Task1.1] requires 4 days
[Task1.2] starts at [Task1]'s end and requires 7 da
[Task2] requires 5 days
then [Task2.1] requires 4 days
[MaxTaskEnd] happens at [Task1.1]'s end
[MaxTaskEnd] happens at [Task1.2]'s end
[MaxTaskEnd] happens at [Task2.1]'s end
"""
)
```



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



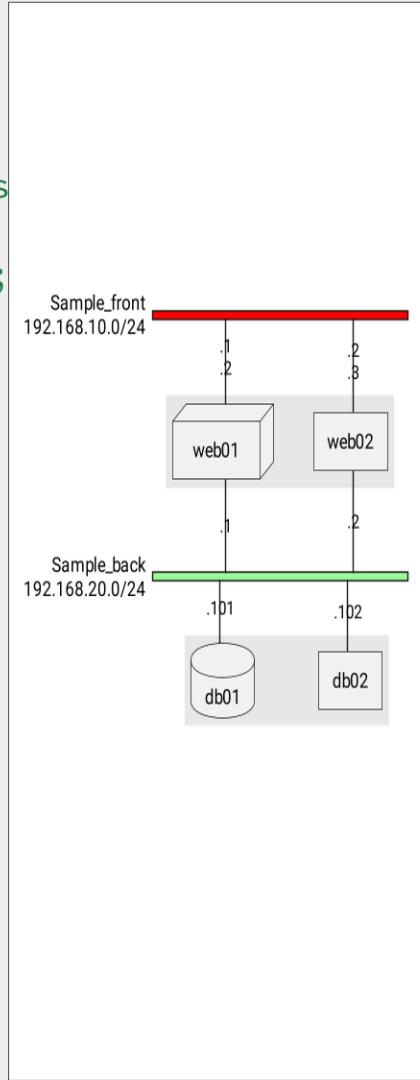
Network Diagrams



```
nwdiag(")

// define group outs
group {
    color = "#FFAAAA";
    web01;
    web02;
    db01;
}
network dmz {
    web01;
    web02;
}
network internal {
    web01;
    web02;
    db01;
    db02;
}

")
```



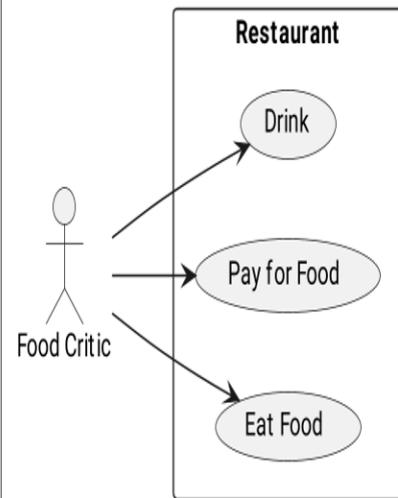
```
nwdiag("

network Sample_front {
    address = "192.168.10.0/24";
    color = "red";
}
// define group
group web {
    web01 [address = "192.168.10.1"];
    web02 [address = "192.168.10.2"];
}
network Sample_back {
    address = "192.168.20.0/24";
    color = "palegreen";
}
web01 [address = "192.168.20.1"];
web02 [address = "192.168.20.2"];
}

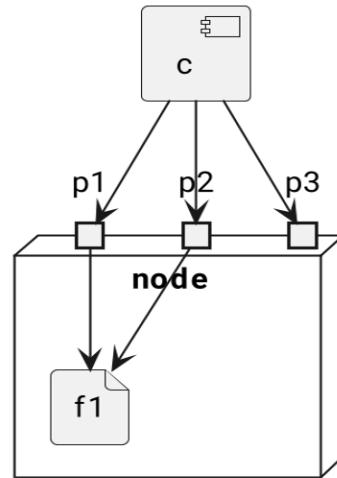
")
```



UML Use Case



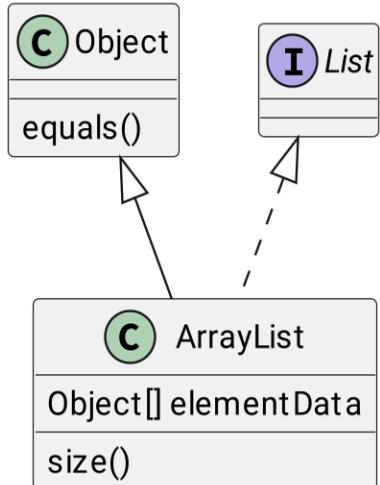
```
uml(
"""
left to right direction
actor "Food Critic" as
rectangle Restaurant {
    usecase "Eat Food" as UC1
    usecase "Pay for Food" as UC2
    usecase "Drink" as UC3
}
fc --> UC1
fc --> UC2
fc --> UC3
"""
)
```



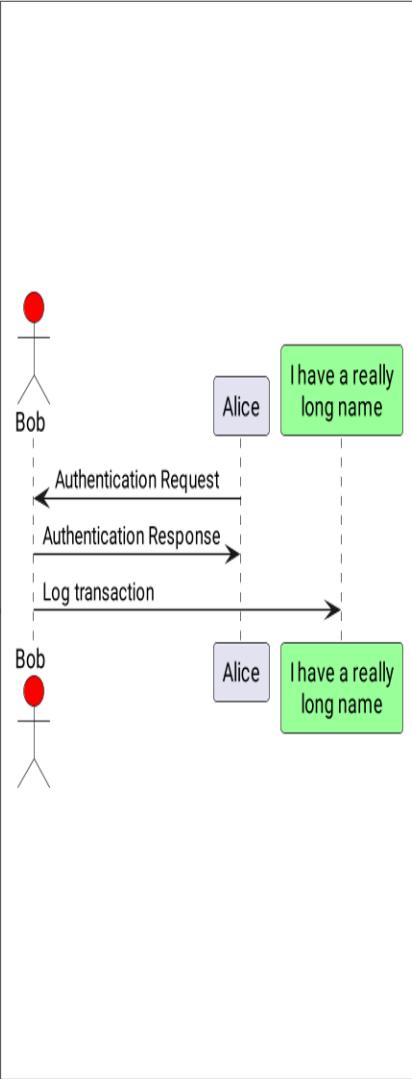
```
uml(
"""
[c]
node node {
    port p1
    port p2
    port p3
    file f1
}
c --> p1
c --> p2
c --> p3
p1 --> f1
p2 --> f1
"""
)
```



UML Classes



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



```
uml( """  
actor Bob #red  
' The only difference b  
' and participant is the  
participant Alice  
participant "I have a r  
' You can also declare  
participant L as "I  
' /  
Alice->Bob: Authentication  
Bob->Alice: Authentication  
Bob->L: Log transaction  
""")
```



Wireframe Diagrams

Login	<input type="text" value="MyName"/>
Password	<input type="password" value="****"/>
<input type="button" value="Cancel"/>	<input type="button" value="OK"/>

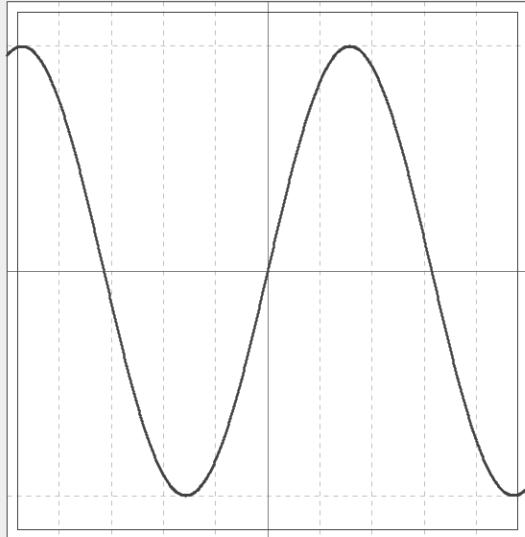
```
wireframe(  
    """  
        Login      | "MyName"  
        Password   | "****"  
        [Cancel]  | [  OK  ]  
    """  
)
```

General	<input type="checkbox"/>
Fullscreen	<input type="checkbox"/>
Behavior	<input type="checkbox"/>
Saving	<input type="checkbox"/>
Open image in:	<input type="button" value="Smart Mode ▾"/>
<input checked="" type="checkbox"/> Smooth images when zoomed	
<input checked="" type="checkbox"/> Confirm image deletion	
<input type="checkbox"/> Show hidden images	
<input type="button" value="Close"/>	

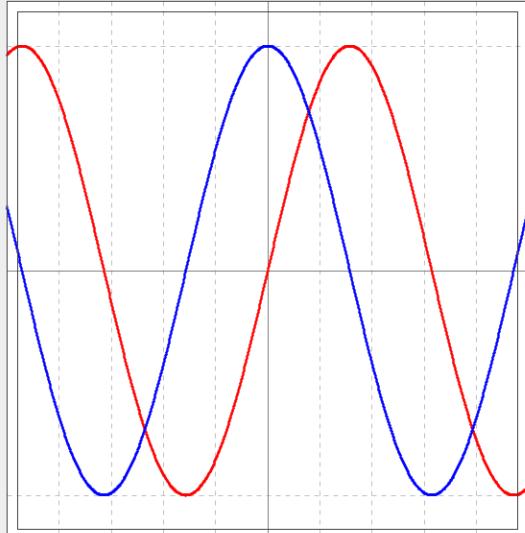
```
wireframe(  
    """  
        {+  
        /<b>General | Fullscr  
        {  
        { Open image in: | ^Sma  
        [X] Smooth images when  
        [X] Confirm image delet  
        [ ] Show hidden images  
        }  
        [Close]  
        }  
    """  
)
```



Plot lines



```
plot2d(xmin:-5 , xmax: 5){  
    curve {  
        f(x):sin(x)  
    }  
}
```



```
plot2d(xmin:-5 , xmax: 5){  
    curve{  
        f(x):sin(x)  
        title:'sin'  
        color:red  
    }  
    curve{  
        f(x):cos(x)  
        title:'cos'  
        color:blue  
    }  
}
```



Test me



open the following file

`<ntexup_gitroot>/documentation/ntexup-doc/02-pages/9999-conclusion/0110`

and try to write some things there...

Thank you

taha.bensalah@gmail.com