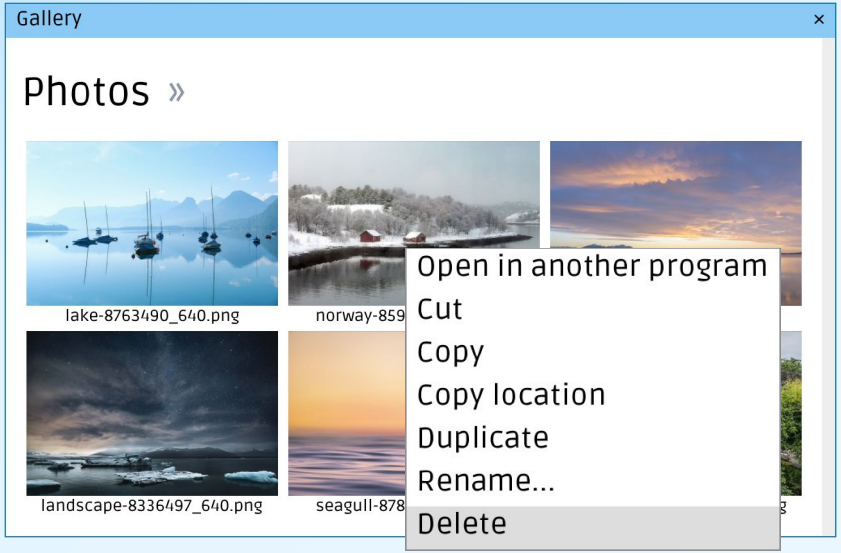
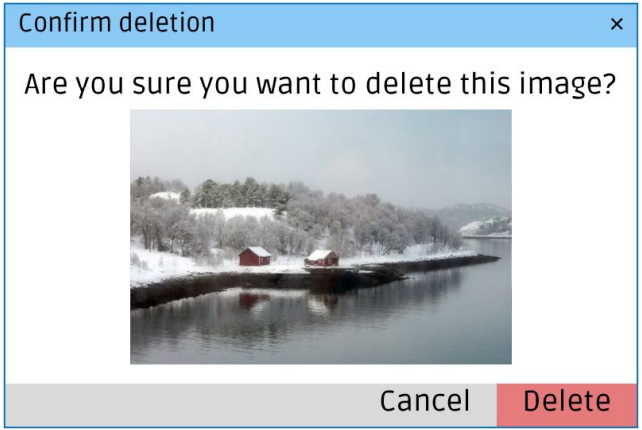
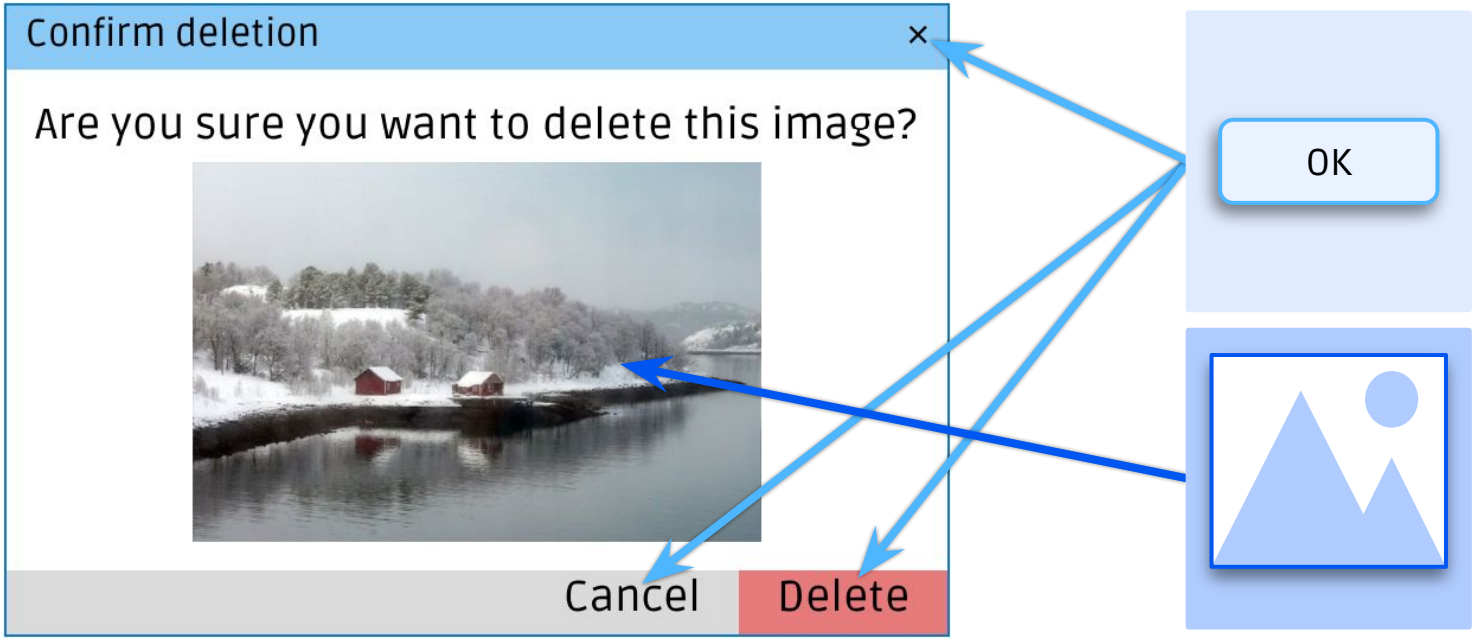
Exploring declarative capabilities of D

User interfaces





Reusable components


Confirm deletion
×
Are you sure you want to delete this image?
Cancel
Delete
OK

Combining components

- Column

Combining components

- Column
  - Text

Combining components

- Column
  - Text
  - Row
    - Text

Combining components

- Column
  - Text
  - Row
    - Text
    - Column
      - Text
      - Button
      - Text

```
auto heading = new Label;
heading.text = "...";

auto leftLabel = new Label;
leftLabel.text = "...";
leftLabel.layout.expand = 1;

auto innerLabel = new Label;
innerLabel.text = "...";

auto button = new Button;
button.text = "OK";
button.pressed = &onClick;

auto column = new Frame;
column.layout.expand = 1;
column.layout.nodeAlign = NodeAlign.fill;
column ~= label1;
column ~= label2;

auto row = new Frame;
row.isHorizontal = true;
row.layout.expand = 1;
row.layout.nodeAlign = NodeAlign.fill;
row ~= leftLabel;
row ~= column;

auto root = new Frame;
root.layout.nodeAlign = NodeAlign.fill;
root ~= heading;
root ~= row;

return root;
```

Coding it in

```
auto heading = new Label;
heading.text = "...";

auto leftLabel = new Label;
leftLabel.text = "...";
leftLabel.layout.expand = 1;

auto innerLabel = new Label;
innerLabel.text = "...";

auto button = new Button;
button.text = "OK";
button.pressed = &onClick;

auto column = new Frame;
column.layout.expand = 1;
column.layout.nodeAlign = NodeAlign.fill;
column ~= label1;
column ~= label2;

auto row = new Frame;
row.isHorizontal = true;
row.layout.expand = 1;
row.layout.nodeAlign = NodeAlign.fill;
row ~= leftLabel;
row ~= column;

auto root = new Frame;
root.layout.nodeAlign = NodeAlign.fill;
root ~= heading;
root ~= row;

return root;
```

Markup?

```
<?xml version="1.0" encoding="UTF-8"?>
<frame align="fill">
  <label>...</label>
  <frame direction="horizontal" align="fill" expand="1">
    <label expand="1">...</label>
    <frame align="fill" expand="1">
      <label>...</label>
      <button pressed="onClick">OK</button>
    </frame>
  </frame>
</frame>
```

```
auto myFrame = loadXML("frame.xml");
myFrame["onClick"] = delegate() {
    writeln("Pressed!");
};
```

Markup?

```
<?xml version="1.0" encoding="UTF-8"?>
<frame align="fill">
  <label>...</label>
  <frame direction="horizontal" align="fill" expand="1">
    <label expand="1">...</label>
    <frame align="fill" expand="1">
      <label>...</label>
      <button pressed="onClick">OK</button>
    </frame>
  </frame>
</frame>
```

```
auto myFrame = loadXML("frame.xml");
foreach (item; ["a", "b", "c"]) {
    auto listItem = loadXML("listitem.xml");
    listItem["content"].text = item;
    myFrame["list"] ~= listItem;
}
myFrame["onClick"] = delegate() {
    writeln("Pressed!");
};
```

Markup?

```
<?xml version="1.0" encoding="UTF-8"?>
<frame align="fill">
  <label>...</label>
  <frame direction="horizontal" align="fill" expand="1">
    <label expand="1">...</label>
    <frame id="list" align="fill" expand="1">
      <button pressed="onClick">OK</button>
    </frame>
  </frame>
</frame>
```

```
// an oversimplification...
auto myFrame = loadTemplate("frame.xml", [
    "list": ["a", "b", "c"],
    "onClick": delegate() {
        writeln("Pressed!");
    }
]);
```

Markup...?

```
<?xml version="1.0" encoding="UTF-8"?>
<frame align="fill">
  <label>...</label>
  <frame direction="horizontal" align="fill" expand="1">
    <label expand="1">...</label>
    <frame align="fill" expand="1">
      {{ foreach item; list }}
        <label>{item}</label>
      {{ end }}
      <button pressed="onClick">OK</button>
    </frame>
  </frame>
</frame>
```

Rewind...

```
int[3] array;
array[0] = 1;
array[1] = 2;
array[2] = 3;
```

Rewind...

```
int[3] array;
array[0] = 1;
array[1] = 2;
array[2] = 3;
```

```
int[3] array = [1, 2, 3];
```

```
auto heading = new Label;
heading.text = "...";

auto leftLabel = new Label;
leftLabel.text = "...";
leftLabel.layout.expand = 1;

auto innerLabel = new Label;
innerLabel.text = "...";

auto button = new Button;
button.text = "OK";
button.pressed = &onClick;

auto column = new Frame;
column.layout.expand = 1;
column.layout.nodeAlign = NodeAlign.fill;
column ~= label1;
column ~= label2;

auto row = new Frame;
row.isHorizontal = true;
row.layout.expand = 1;
row.layout.nodeAlign = NodeAlign.fill;
row ~= leftLabel;
row ~= column;

auto root = new Frame;
root.layout.nodeAlign = NodeAlign.fill;
root ~= heading;
root ~= row;

return root;
```

Forward.

```
return vframe(
    .layout!"fill",
    label("..."),
    hframe(
        .layout!(1, "fill"),
        label(
            .layout!1,
            "...",
        ),
        vframe(
            .layout!(1, "fill"),
            label(progress ~ " completed"),
            button("OK", &onClick),
        ),
    )
);
```

Going declarative

```
new Frame(
     new Label("Frame content"),
     new Label("goes here..."),
     new Button("OK", &onClick),
);
new ImageView("res/image.png");
```

Variants...

`new Frame()`	`Frame.vertical()` `Frame.horizontal()`

`new Button()`
`new FrameButton()`
	`new Button()`
	`FrameButton.vertical()`
	`FrameButton.horizontal()`

`new NumberInput!T()`	`new NumberInput!T()`
	`new NumberInput!int()`
	`new NumberInput!float()`

`new TextInput()`	`TextInput.line()` `TextInput.multiline()`

Variants.

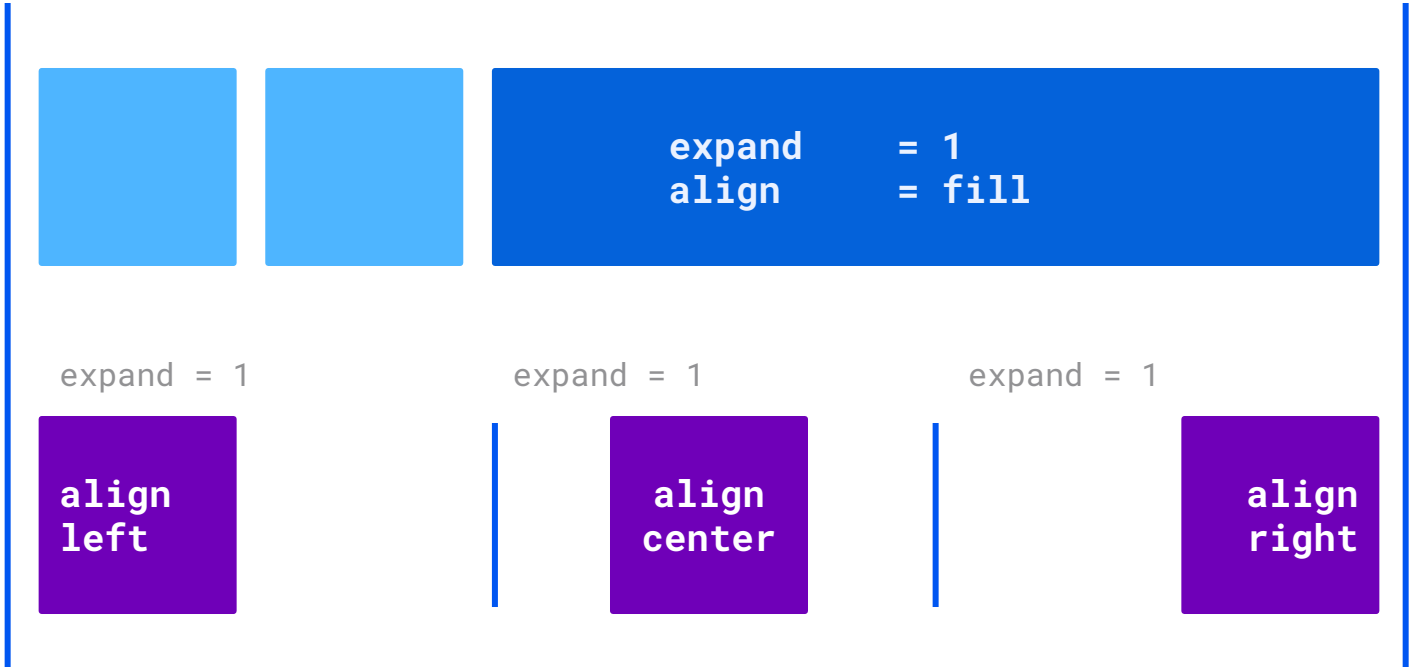
`new Frame()`	`vframe()` `hframe()`

`new Button()`
`new FrameButton()`
	`button()`
	`vframeButton()`
	`hframeButton()`

`new NumberInput!T()`	`numberInput!T()`
	`intInput()`
	`floatInput()`

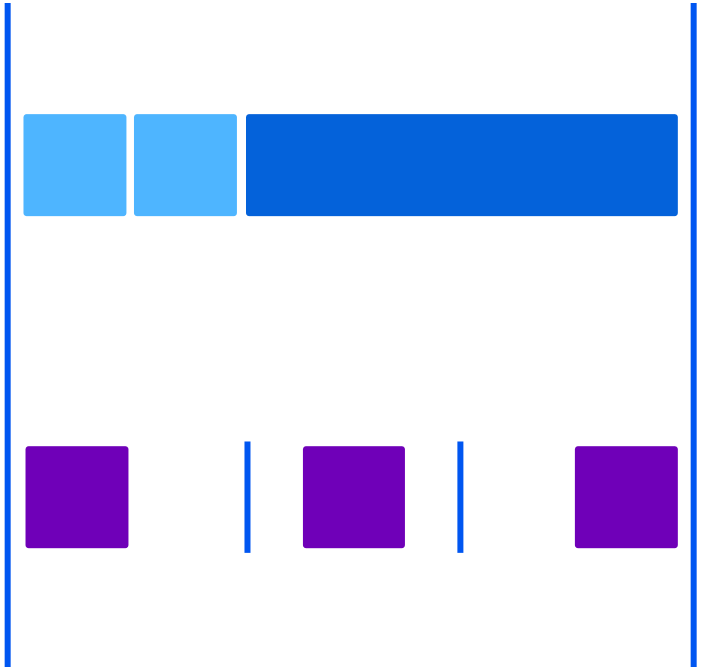
`new TextInput()`	`lineInput()` `textInput()`

Properties.


expand = 1
align = fill
expand = 1
expand = 1
expand = 1
align left
align center
align right

```
hframe(
     .layout!"fill",
     box(),
     box(),
     box(
          .layout!(1, "fill"),
     ),
);

hframe(
     .layout!"fill",
     box(.layout!(1, "start")),
     box(.layout!(1, "center")),
     box(.layout!(1, "end")),
);
```



```
sizeLock!hframe(
     .myTheme,
     .layout!(1, "fill"),
     .tags!(tag1, tag2, tag3),
     .hidden,
     .disabled,
     .acceptDrop,
     .sizeLimit(100, 200),
     label("Hello, World!")
);
```

Node builder

```
sizeLock!hframe(
     .myTheme,
     .layout!(1, "fill"),
     .tags!(tag1, tag2, tag3),
     .hidden,
     .disabled,
     .acceptDrop,
     .sizeLimit(100, 200),
     label("Hello, World!")
);
```

`sizeLock!hframe` — node builder
`hidden`, etc. — node parameter

Node builder

```
sizeLock!hframe(
     .myTheme,
     .layout!(1, "fill"),
     .tags!(tag1, tag2, tag3),
     .hidden,
     .disabled,
     .acceptDrop,
     .sizeLimit(100, 200),
     label("Hello, World!")
);
```

```
auto hidden(bool value = true) {

    static struct Hidden {
        bool value;

        void apply(Node node) {
            node.isHidden = value;
        }

    }

    return Hidden(value);

}
```

Node builder

```
sizeLock!hframe(
     .myTheme,
     .layout!(1, "fill"),
     .tags!(tag1, tag2, tag3),
     .hidden,
     .disabled,
     .acceptDrop,
     .sizeLimit(100, 200),
     label("Hello, World!")
);
```

```
auto node = new SizeLock!Frame(
    label("Hello, World!")
);
node.isHorizontal = true;
myTheme.apply(node);
layout!(1, "fill").apply(node);
tags!(tag1, tag2, tag3).apply(node);
hidden.apply(node);
disabled.apply(node);
acceptDrop.apply(node);
sizeLimit(100, 200).apply(node);
```

~~UFCS~~

Node builder

```
sizeLock!hframe(
     .myTheme,
     .layout!(1, "fill"),
     .tags!(tag1, tag2, tag3),
     .hidden,
     .disabled,
     .acceptDrop,
     .sizeLimit(100, 200),
     label("Hello, World!")
);
```

UFCS

```
sizeLock!hframe(
    label("..."),
    hframe(
        label(
            "...")
            .layout!1,
        vframe(
            label("..."),
            label("..."))
            .layout!(1, "fill"))
        .layout!(1, "fill"),
    hframe(
        label(
            "...")
            .layout!1,
```

Node builder

```
sizeLock!hframe(
    .myTheme,
    .layout!(1, "fill"),
    .tags!(tag1, tag2, tag3),
    label("..."),
    hframe(
        .layout!(1, "fill"),
        label(
            .layout!1,
            "...",
        ),
        vframe(
            .layout!(1, "fill"),
            label("..."),
```

Revealing the magic

```
alias vframe = nodeBuilder!Frame;
alias hframe = nodeBuilder!(Frame, (a) {
    a.isHorizontal = true;
});

class Frame : Space {
    ...
}
```

Recently renamed: `nodeBuilder` used to be called "`simpleConstructor`"

```
alias vframe = nodeBuilder!Frame;
enum nodeBuilder(T, alias fun = "a")
    = NodeBuilder!(T, fun).init;
```

```
Type = Frame
```

```
alias vframe = nodeBuilder!Frame;
enum nodeBuilder(T, alias fun = "a")
    = NodeBuilder!(T, fun).init;
struct NodeBuilder(T, alias fun = "a") {
    alias Type = T;
    ...
}
```

```
struct NodeBuilder(T, alias fun = "a") {
    alias Type = T;
    alias initializer = unaryFun!fun;

    Type opCall(Args...)(Args args) {
        ...
    }

}
```

```
Type = Frame
opCall(Args...)
```

```
vframe(
    .myTheme,
    .layout!"fill",
    label("Hello, "),
)
```

```
struct NodeBuilder(T, alias fun = "a") {
    alias Type = T;
    alias initializer = unaryFun!fun;

    Type opCall(Args...)(Args args) {

        enum paramCount = leadingParams!Args;
        auto result = new Type(args[paramCount..$]);
        initializer(result);
```

```
Type = Frame
opCall(Args...)
```

```
vframe(
    label("Hello, "),
    label("World! "),
)



// paramCount == 0
```

```
vframe(
    .myTheme,
    .layout!"fill",
    label("Hello, "),
    label("World! "),
)

// paramCount == 2
```

```
vframe(
    .layout!"fill",
    label("..."),
    .myTheme,
)


// paramCount == 1
```

```
Type = Frame
opCall(Args...)
leadingParams(Args...)
```

```
struct NodeBuilder(T, alias fun = "a") {
    alias Type = T;
    alias initializer = unaryFun!fun;
    Type opCall(Args...)(Args args) { ... }

    static int leadingParams(Args...)() {

        foreach (i, Arg; Args) {
            if (!isNodeParam!(Arg, T)) {
                return i;
            }
        }
        return Args.length;

    }

}
```

```
vframe(
    .myTheme,           //  isNodeParam!Theme   0
    .layout!"fill",     //  isNodeParam!Layout  1
    label("Hello, "),   // !isNodeParam!Label   2
    vframe(),
)
```

```
Type = Frame
opCall(Args...)
leadingParams(Args...)
```

```
static int leadingParams(Args...)() {

    foreach (i, Arg; Args) {
        if (!isNodeParam!(Arg, T)) {
            return i;
        }
    }
    return Args.length;

}

}
```

```
struct NodeBuilder(T, alias fun = "a") { ... }

enum isNodeParam(T, NodeType = Node)
    = __traits(compiles, T.init.apply(NodeType.init));
```

```
Type = Frame
opCall(Args...)
leadingParams(Args...)
isNodeParam!(Arg, Type)
```

```
struct NodeBuilder(T, alias fun = "a") { ... }

enum isNodeParam(T, NodeType = Node)
    = __traits(compiles, T.init.apply(NodeType.init));
```

```
Type = Frame
opCall(Args...)
leadingParams(Args...)
isNodeParam!(Arg, Type)
```

```
.myTheme.apply(vframe())
.layout!"fill".apply(vframe())
```

```
label().apply(vframe())
.sizeLimit(100).apply(vframe())
```

```
struct NodeBuilder(T, alias fun = "a") {
    alias Type = T;
    alias initializer = unaryFun!fun;

    Type opCall(Args...)(Args args) {

        enum paramCount = leadingParams!Args;
        auto result = new Type(args[paramCount..$]);
        initializer(result);
        foreach (param; args[0..paramCount]) {
            param.apply(result);
        }
        return result;
    }

}
```

```
Type = Frame
opCall(Args...)
```

```
Type = Frame
opCall(Args...)
```

```
struct NodeBuilder(T, alias fun = "a") {
    alias Type = T;
    alias initializer = unaryFun!fun;

    Type opCall(Args...)(Args args) {

        enum paramCount = leadingParams!Args;
        auto result = new Type(args[paramCount..$]);
        initializer(result);
        foreach (param; args[0..paramCount]) {
```

```
alias vframe = nodeBuilder!Frame;
alias hframe = nodeBuilder!(Frame, (a) {
    a.directionHorizontal = true;
});
```

Thank you!

```
dub init -t fluid
```

/samerion

Slides by Artha
Photo credits:

- https://pixabay.com/photos/lake-lake-wolfgang-salzkammergut-8763490/
- https://pixabay.com/photos/norway-winter-snow-atlantic-8593725/
- https://pixabay.com/photos/sunset-dusk-mountains-clouds-8516639/
- https://pixabay.com/photos/landscape-sea-ice-darling-clouds-8336497/
- https://pixabay.com/photos/seagull-waves-marine-sea-sunset-8781110/
- https://pixabay.com/photos/landscape-roman-bridge-river-8487906/