Visual Interface Dialect

Table of Contents

1. Úvod

VID (Visual Interface Dialect) je jednoduchý dialekt (DSL) pro popis grafického uživatelského rozhranní nad systémem View.

VID umožňuje určení polohy každého grafického elementu a to třemi způsoby:

- horizontální či vertikální uspořádání
- umístění elementu do mřížky
- absolutní zadání pozice

VID automaticky vytvoří kontejnerový piškot pro umístění zadaných piškotů. Implicitně je kontejnerový piškot typu `window`.

Kód VIDu je vyhodnocen funkcí `layout` (jež je interně volána funkcí `view`). Kód je poté kompilován do stromu piškotů, vhodného pro přímé zobrazení.

NOTE

- Použijte `help view` a `help layout` v konzole Redu k získání přehledné informace.
- Typy piškotů nepatří mezi datové typy!

2. Struktura kódu

Typický blok kódu VID má následující strukturu:

```
[
    <nastavení kontejneru>
    <popis rozvržení>
]
```

- **nastavení kontejneru**: údaje, které ovlivňují objekt kontejneru (jímž může být panel nebo window).
- **popis rozvržení**: poziční příkazy, definice stylů a popisy piškotů.

NOTE Všechny sekce jsou nepovinné; blok VIDu nemá předepsaný obsah.

3. Nastavení kontejnerů

NOTE Klíčové slovo `react` keyword může být použito při nastavení kontejneru i při výběru možností pro `face` - viz podrobný popis v odstavci zde.

3.1. title

Syntaxe

```
title <text>

<text> : text titulku (string!).
```

Popis

Určí titulek kontejnerového piškotu.

3.2. size

Syntaxe

```
size <value>

<value> : šířka a výška v pixelech (pair!).
```

Popis

Nastavuje velikost kontejnerového piškotu. Není-li velikost explicitně zadána, je automaticky spočtena tak aby obsáhla vložené elementy.

3.3. backdrop

Syntaxe

```
backdrop <color>

<color> : název nebo hodnota barvy (word! tuple! issue!).
```

Popis

Nastavuje barvu pozadí kontejnerového piškotu.

Příklad:

```
view layout [
    title "Frontální útok"
    size 300x60
    backdrop 205.180.200
    button "Nazdar"
]
```

3.4. Definice aktérů

V této oblasti kódu je také možné definovat aktéry kontejneru - viz sekci [Actors].

4. Rozvržení piškotů

VID umisťuje piškoty do piškotu window podle jednoduchých pravidel:

- směr řazení piškotů může být horizontální nebo vertikální
- piškoty jsou v daném směru umisťovány jeden za druhým se zadaným odstupem

Implicitní hodnoty:

- počátek (origin): `10x10`
- mezera (space): `10x10`
- směr (direction): `across`
- alignment: `top`

Takto jsou piškoty rozmisťovány v režimu `across`:

[across] | *across.png*

Takto jsou piškoty rozmisťovány v režimu below (s použitím implicitního přiřazení `left`):

[below] | *below.png*

4.1. across

Syntaxe

```
across <alignment>

<alignment> : (optional) possible values: top | middle | bottom.
```

Popis

Rozmisťování probíhá v horizontálním směru zleva doprava. Implicitní přiřazení piškotů v řadě (`top`) lze změnit modifikátorem přiřazení.

4.2. below

Syntaxe

```
below <alignment>

<alignment> : (optional) possible values: left | center | right.
```

Popis

Rozmisťování probíhá ve vertikálním směru shora dolů. Implicitní přiřazení piškotů ve sloupci (`left`) lze změnit modifikátorem přiřazení.

4.3. return

Syntaxe

```
return <alignment>

<alignment> : (optional) possible values: left | center | right | top | middle |
bottom.
```

Popis

Přesouvá pozici na další řádek nebo sloupec piškotů v závislosti na aktuálním směru rozmisťování. Implicitní přiřazení piškotů v řadě či sloupci lze změnit modifikátorem přiřazení.

4.4. space

Syntaxe

```
space <offset>

<offset> : nová hodnota mezery (pair!).
```

Popis

Udává hodnotu odstupu pro nově umisťované piškoty.

4.5. origin

Syntaxe

```
origin <offset>

<offset> : nová hodnota počátku (pair!).
```

Popis

Udává novou pozici počátku, relativně k rohu kontejnerového piškotu.

4.6. at

Syntaxe

```
at <offset>
at <expr>

<offset> : pozice dalšího piškotu (pair!).
<expr>   : výraz, vracející hodnotou typu pair! jako pozici
```

Popis

Umisťuje další piškot do absolutně zadané pozice. Tento poziční režim se vztahuje pouze k následujícímu piškotu a nemění zadaný způsob umístění pro piškoty další.

4.7. pad

Syntaxe

```
pad <offset>

<offset> : relativní odsazení (pair!).
```

Popis

Upravuje pozici piškotu o relativní odsazení (offset). Všechny následující piškoty v řadě či sloupci se příslušně posunou také.

4.8. do

Syntaxe

```
do <body>

<body> : prováděný kód (block!).
```

Popis

Vyhodnotí blok regulérního kódu Redu pro následné "last-minute" inicializační použití. Blok těla je vázán na piškot kontejneru (okno nebo panel), takže je možný přímý přístup k pišotu kontejneru. Na kontejner samotný lze odkazovat klíčovým slovem `self`.

5. Dodatečné styly

View engine poskytuje mnoho vestavěných piškotů. Dialekt VID je rozšiřuje definováním dalších obecně používaných stylů s přiřazenými klíčovými slovy. Lze je použít se stejnými volbami jako jejich výchozí typ piškotu. Mohou byt také redefinovány použitím příkazu `style`.

5.1. h1

Styl `H1` je typu `text` s velikostí fontu 32.

5.2. h2

Styl `H2` je typu `text` s velikostí fontu 26.

5.3. h3

Styl `H3` je typu `text` s velikostí fontu 22.

5.4. h4

Styl `H4` je typu `text` s velikostí fontu 17.

5.5. h5

Styl `H5` je typu `text` s velikostí fontu 13.

5.6. box

Styl `box` je typu `base` s implicitně nastavenou transparentní barvou.

5.7. image

Styl `image` je typu `base` s implicitní velikostí 100x100.

6. Definice piškotů

Piškot lze vložit do aktuální pozice rozvržení (layout) jednoduše uvedením jména existujícího typu piškotu nebo dostupného stylu.

Syntaxe

```
<name>: <type> <options>

<name>    : název nového komponentu (set-word!).
<type>    : platný typ piškotu nebo název stylu (word!).
<options> : seznam možností
```

Zadaný název odkazuje na objekt typu `face!`, vytvořený dialektem VID z popisu piškotu.

Pro každý styl nebo typ piškotu jsou k dispozici implicitní hodnoty, pročež lze vytvořit nový piškot bez jakýchkoliv specifikací. Případně potřebné specifikace se dělí do těchto skupin:

- Keywords - klíčová slova
- Datatypes- datové typy
- Actors - aktéři

Všechny parametry lze zadávat v libovolném pořadí za názvem piškotu nebo stylu. Nový název piškotu nebo klíčové slovo rozmístění (layoutu) označuje konec seznamu parametrů (options) pro daný piškot.

NOTE Slovo `window` nemůže být použito jako typ piškotu.

6.1. Klíčová slova

6.1.1. left

Syntaxe

```
left
```

Popis

Zarovná text piškotu k levému okraji.

6.1.2. center

Syntaxe

```
center
```

Popis

Vystředí text piškotu

6.1.3. right

Syntaxe

```
right
```

Popis

Zarovná text pišotu k pravému okraji.

6.1.4. top

Syntaxe

```
top
```

Popis

Zarovná text pišotu k hornímu okraji.

6.1.5. middle

Syntaxe

```
middle
```

Popis

Umístí text piškotu vertikálně doprostřed.

6.1.6. bottom

Syntaxe

```
bottom
```

Popis

Zarovná text piškotu k dolnímu okraji .

6.1.7. bold

Syntaxe

```
bold
```

Popis

Nastaví styl textu na `bold`.

6.1.8. italic

Syntaxe

```
italic
```

Popis

Nastaví styl textu na `italic`.

6.1.9. underline

Syntaxe

```
underline
```

Popis

Nastaví styl textu na `underline`.

6.1.10. extra

Syntaxe

```
extra <expr>

<expr> : jakákoli hodnota nebo výraz Redu (any-type!).
```

Popis

Nastaví aspekt piškotu `extra` na novou hodnotu.

6.1.11. data

Syntaxe

```
data <list>
data <expr>

<list> : literálový seznam položek nebo výraz Redu (block!).
<expr> : výraz, vracející seznam jako block!
```

Popis

Nastaví aspekt `data` piškotu na seznam hodnot. Formát seznamu závisí na požadavcích typu piškotu.

6.1.12. draw

Syntaxe

```
draw <commands>
draw <expr>

<commands> : literálový seznam příkazů nebo výraz Redu(block!).
<expr>     : výraz, vracející blok příkazů (block!).
```

Popis

Nastaví aspekt `draw` piškotu na seznam příkazů dialektu Draw. Viz link:draw.adoc [Draw dialect].

6.1.13. font

Syntaxe

```
font <spec>

<spec> : zadání platného fontu (block! object! word!).
```

Popis

Nastaví aspekt `font` piškotu na nový objekt `font!`. Objekt font! je popsán zde.

NOTE	Je možné použít `font` spolu s jinými souvisejícími parametry. VID je sloučí dohromady s prioritou posledně zadaného parametru.

6.1.14. para

Syntaxe

```
para <spec>

<spec> : určení platného objektu para (block! object! word!).
```

Popis

Nastaví aspekt `para` novému objektu `para!`. Objekt para! je popsán zde.

NOTE	Je možné použít `para` spolu s jinými souvisejícími parametry. VID je sloučí dohromady s prioritou posledně zadaného parametru.

6.1.15. wrap

Syntaxe

```
wrap
```

Popis

Při zobrazení omezit délku textového řádku.

6.1.16. no-wrap

Syntaxe

```
no-wrap
```

Popis

Neomezovat délku zobrazeného textu.

6.1.17. font-size

Syntaxe

```
font-size <pt>

<pt> : velikost fontu v bodech (integer! word!).
```

Popis

Nastaví velikost fontu pro zobrazovaný text piškotu.

6.1.18. font-color

Syntaxe

```
font-color <value>

<value> : barva fontu (tuple! word! issue!).
```

Popis

Nastaví barvu aktuálního fontu pro text piškotu.

6.1.19. font-name

Syntaxe

```
font-name <name>

<name> : platný název dostupného fontu (string! word!).
```

Popis

Nastaví název fontu v piškotu.

6.1.20. react

Toto klíčové slovo lze použít jako možnost piškotu i ve smyslu globálním. Lze použít libovolný počet instancí slova `react`.

Syntaxe

```
react [<body>]
react later [<body>]

<body> : regulerní kód Redu (block!).
```

Popis

Vytvoří nový reaktor z těla bloku. Je-li `react` použito jako možnost (option) piškotu, může tělo bloku odkazovat na aktuální piškot s použitím slova `face`. Je-li slovo `react` použito globálně, musí být cílové piškoty volatelné jménem.

Nepovinné klíčové slovo `later` přeskočí první událost reakce bezprostředně po provedení `těla` bloku.

NOTE	Reaktory jsou součástí reaktivního programování ve View, jehož dokumentace se připravuje. Stručně řečeno, tělo bloku může popisovat jeden či více vztahů mezi vlastnostmi piškotů a to s použitím cest. Nastavení vlastnosti piškotu je zpracováno jako cíl (target) reaktoru (aktualizovaný piškot), zatímco cesta, vedoucí k vlastnosti piškotu je zpracována jako zdroj (source) reaktoru (změna zdroje spustí aktualizaci kódu reaktoru).

6.1.21. loose

Syntaxe

```
loose
```

Popis

Umožňuje tažení piškotu levým tlačítkem myši.

6.1.22. all-over

Syntaxe

```
all-over
```

Popis

Nastaví flag `all-over`, který povoluje příjem všech "myších" událostí `over`.

6.1.23. hidden

Syntaxe

```
hidden
```

Popis

Činí piškot neviditelným.

6.1.24. disabled

Syntaxe

```
disabled
```

Popis

Vypíná aktivitu piškotu (piškot nezpracovává žádnou událost).

6.1.25. password

Syntaxe

```
password
```

Popis

Skryje vstup uživatele v textovém poli.

6.1.26. tri-state

Syntaxe

```
tri-state
```

Popis

Umožní tří-stavový režim zatržítka (check box).

6.1.27. select

Syntaxe

```
select <index>

<index> : index vybrané položky (integer!).
```

Popis

Sets the `selected` facet of the current face. Used mostly for lists to indicate which item is pre-selected.

6.1.28. focus

Syntaxe

```
focus
```

Popis

Dodává zaměření (focus) aktuálnímu piškotu při prvním zobrazení okna. Zaměření lze udělit pouze jednomu piškotu. Je-li použito několik voleb `focus` pro různé piškoty, dostane se zaměření jen tomu poslednímu.

6.1.29. hint

Syntaxe

```
hint <message>

<message> : text návodu (string!).
```

Popis

Poskytuje text návodu uvnitř polí piškotů, které dosud nemají žádný obsah. Tento text zmizí při

zadání nového obsahu (akcí uživatele nebo nastavením parametru `face/text`).

6.1.30. rate

Syntaxe

```
rate <value>
rate <value> now

<value>: trvání nebo frekvence (integer! time!).
```

Popis

Nastaví časovač piškotu pro trvání (time!) nebo frekvenci (integer!). Při každém tiku časovače je generována událost `time` piškotu. Je-li použita volba `now`, je první časová událost generována okamžitě.

6.1.31. default

Syntaxe

```
default <value>

<value>: impmlicitní hodnota aspektu `data` (any-type!).
```

Popis

Definuje implicitní hodnotu aspektu `data`, když konverze aspektu `text` vrací `none`. Tato implicitní hodnota je uložena v aspektu `options` jako pár key/value.

> **NOTE** aktuálně používáno pouze u piškotů `text` a `field`.

6.1.32. with

Syntaxe

```
with <body>

<body>: blok kódu vázaný na aktuální piškot (block!)
```

Popis

Vyhodnocuje blok kódu, vázaný na aktuálně definovaný piškot. Umožňuje přímé nastavení polí piškotu, potlačujíc jiná nastavení VID.

6.2. Datové typy

Kromě klíčových slov je možné zadat nastavení piškotů s použitím literálních hodnot následujících typů:

Datatype	Purpose
integer!	Určuje šířku piškotu. U panelů indikuje počet řad nebo sloupců v uspořádání (layout) v závislosti na aktuálním směru.
pair!	Určuje šířku a výšku piškotu.
tuple!	Určuje barvu pozadí piškotu (kde je použitelné).
issue!	Určuje barvu pozadí piškotu pomocí hexadecimálního zápisu (#rgb, #rrggbb, #rrggbbaa).
string!	Určuje text, který má být piškotem zobrazen.
date!	Nastavuje aspekt `data` (užitečné pro typ `calendar`).
percent!	Nastavuje aspekt `data` piškotu (užitečné pro typy `progress` a `slider`).
logic!	Nastavuje aspekt `data` piškotu (užitečné pro typy `toggle`, `check` a `radio`).
image!	Určuje obrázek pro pozadí piškotu(tam, kde je použitelné).
url!	Načte zdroj, na nějž ukazuje URL a poté jej provede.
block!	Udává akci pro implicitní událost piškotu. U panelů určuje jejich obsah.
get-word!	Jako aktéra používá existující funkci.
char!	*(vyhraženo pro budoucí použití).*

6.3. Aktéři

Aktér (actor) může být připojen (hooked) k piškotu určením literálové hodnoty bloku nebo názvu aktéra následovaného hodnotou bloku.

Syntaxe

```
<actor>
on-<event> <actor>

<actor> : tělo bloku aktéra nebo odkaz na aktéra (block! get-word!).
<event> : platný název události (word!).
```

Popis

Je možné určit aktéra zjednodušeným způsobem poskytnutím pouze bloku jeho těla. Následně je sestavena funkce aktéra a přidána do aspektu `actor` piškotu. Takto lze určit několik aktérů.

Úplná specifikace funkce vytvářeného aktéra je:

```
func [face [object!] event [event! none!]][...body...]
```

Platný seznam názvů událostí lze nalézt zde.

Je-li zadán blok nebo get-word bez předložky s názvem aktéra, je implicitní aktér pro typ piškotu vytvořen podle definicí zde

Aktér může být rovněž definován mimo VID a odkaz na něj zadán jako get-word argument za tečkou (dot).

7. Panely

Je možné definovat dětské panely pro seskupování piškotů a případně na ně aplikovat specifické styly. Není-li specificky určena, je velikost nového panelu automaticky spočítána podle velikosti jeho obsahu.

Piškoty typu panel ze systému View jsou ve VID podporovány se specifickou syntaxí:

7.1. panel

Syntaxe

```
panel <options> [<content>]

<options> : seznam s nastavením panelu
<content> : popis obsahu VID panelu (block!).
```

Popis

Vytvoří dětský panel uvnitř aktuálního kontejneru, jehož obsah je další blok VID. Kromě dalších opcí piškotu lze zadat celočíselný dělitel, ustavujíc tak uspořádání do mřížky (grid-mode layout):

- je-li zvolený směr `across`, představuje dělitel počet sloupců.
- je-li zvolený směr `below`, představuje dělitel počet řad.

7.2. group-box

Syntaxe

```
group-box <divider> <options> [<body>]

<divider> : zvolený počet řádků nebo sloupců (integer!).
<options> : seznam nastavení pro panel.
<body>    : popis obsahu VID panelu (block!).
```

Popis

Vytvoří panel typu group-box uvnitř aktuálního kontejneru, kde obsahem je další blok VID. Eventuelně zadaný dělitel ustavuje uspořádání (layout) do mřížky:

- je-li zvolený směr `across`, představuje dělitel počet sloupců.
- je-li zvolený směr `below`, představuje dělitel počet řad.

NOTE	Zadaná hodnota typu `string!` jako opce představuje titulek panelu group-box.

7.3. tab-panel

Syntaxe

```
tab-panel <options> [<name> <body>...]

<options> : zadaný seznam s nastavením panelu.
<name>    : titulek karty (string!).
<body>    : obsah karty (tab) jako popis VID (block!).
```

Popis

Vytvoří `tab-panel` uvnitř aktuálního kontejneru. Specifikační blok musí obsahovat jméno a popis obsahu pro každou kartu (tab). Každé tělo obsahu je nový dětský piškot typu panel, působící jako jakékoliv jiné panely.

8. Určování stylu

8.1. style

Syntaxe

```
style <new> <old> <options>

<new>     : název nového stylu (set-word!).
<old>     : název starého stylu (word!).
<options> : volitelný seznam nastavení pro nový styl.
```

Popis

Nastaví nový styl v aktuálním panelu. Nový styl může být vytvořen z existujících typů piškotů neb z jiných stylů. Nový styl je platný pouze v aktuálním panelu a jeho dětských panelech.

Styly lze kaskádovat od rodičovských panelů k dětským panelům, takže týž styl může být v dětských panelech redefinován nebo rozšířen bez ovlivnění definic v rodičovských panelech.

9. Metriky GUI pro různé platformy

Aby bylo možné vyhovět odlišným GUI požadavkům různých platforem, vkládá VID přepisovací stroj, schopný dynamické úpravy stromu piškotů (face tree) podle zadaných pravidel. Je zařazen jako poslední stupeň procesu VID.

Pravidla pro Windows:

- color-backgrounds: color the background of some colorless faces to match their parent's color
- color-tabpanel-children: Like color-backgrounds, but tab-panel specific
- OK-Cancel: buttons ordering rule, puts Cancel/Delete/Remove buttons last

Pravidla pro macOS:

- adjust-buttons: use standard button sub-classes when buttons are narrow enough
- capitalize: capitalize widget text according to macOS guidelines
- Cancel-OK: buttons ordering rule, puts Ok/Save/Apply buttons last

---

Jednoduchý příklad, který využívá pravidla pro uspořádání tlačítek (buttons) a pro zvětšování písmen:

```
    view [
        text "Name" right 50 field return
        text "Age"  right 50 field return
        button "ok" button "cancel"
    ]
```

Všimněte si textu a uspořádání tlačítek u zobrazení v macOS a ve Windows.

[mac]

[windows]

Popisovaná pravidla pro GUI zajistila, že:

- tlačítka jsou uspořádána podle pravidel jednotlivých platforem, "Ok" poslední v macOS, "Cancel" poslední ve Windows.
- popisky tlačítek jsou řádně 'kapitalizované' v macOS.

Pravidla pro GUI lze vypnout nastavením `system/view/VID/GUI-rules/active?` na `no`.

```
    system/view/VID/GUI-rules/active?: no
```

Pravidla lze odpojit selektivně úpravou obsahu následujících seznamů:

```
system/view/VID/GUI-rules/OS/Windows
== [
    color-backgrounds
    color-tabpanel-children
    OK-Cancel
]
```

```
system/view/VID/GUI-rules/OS/macOS
== [
    adjust-buttons
    capitalize
    Cancel-OK
]
```

Tato procedura umožňuje podřídit se různým požadavkům pro UI bez většího úsilí.