PavlOS Developer Documentation

Content

[Start 1](#_Toc56424647)

[Design 2](#_Toc56424648)

[Overview 2](#_Toc56424649)

[Design demonstration in JSONu 3](#_Toc56424650)

[Detailed description of objects in the page 5](#_Toc56424651)

[Values 5](#_Toc56424652)

[Window 5](#_Toc56424653)

[Text 5](#_Toc56424654)

[Button 5](#_Toc56424655)

[Text box 6](#_Toc56424656)

[Border 6](#_Toc56424657)

[Check box 6](#_Toc56424658)

[Text box multiline 7](#_Toc56424659)

[Web browser 7](#_Toc56424660)

[Styles 8](#_Toc56424661)

[Create a configuration file for styles 8](#_Toc56424662)

[commands in pushers 8](#_Toc56424663)

[Programming 9](#_Toc56424664)

[Overview 9](#_Toc56424665)

[Variable types 9](#_Toc56424666)

[Write variables 9](#_Toc56424667)

[Use of variables 10](#_Toc56424668)

[Library 10](#_Toc56424669)

[Set variable values 10](#_Toc56424670)

[Repeat (for loop) 12](#_Toc56424671)

[Working with the system 12](#_Toc56424672)

[Print to log 12](#_Toc56424673)

# Start

To create apps in PavlOS, you'll need to **have knowledge of JSON** and our Pavl+ **language.** If you do not know how to program in Pavl+ so do not despair, we will teach you.

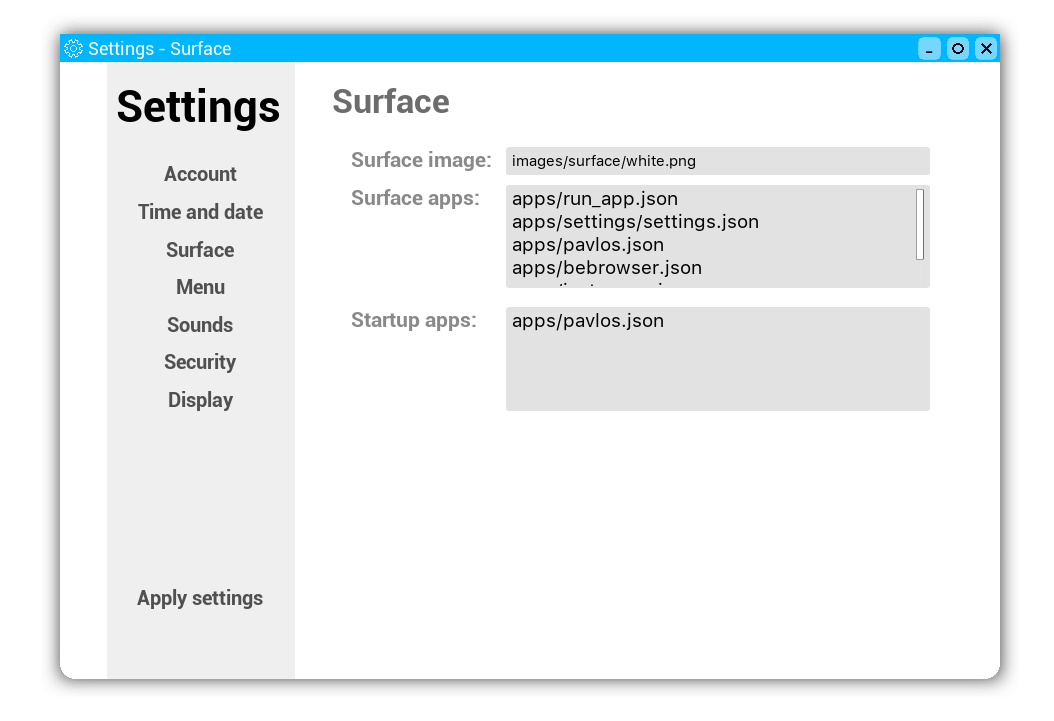
You're going to need a text editor. We recommend **Notepad** or Visual **Studio Code**. Maybe we already have our own editor or compiler for Pavl+ and JSON design at this time, soyou can use it.

# Design

## Overview

The design is created in JSON data files. If you have already created programs in Win UI in XAML, our system in JSON is very similar. In the next part of this chapter, we will show you what creating the design of the application "PavlOS Introduction" looks like. You can also look into application files and look at their structure so that you can see other objects using them for faster understanding.  
  


PavlOS Introduction app



Settings app

Our design at JSON consists of a few sections.

The first object, or "section," means the page number. An application can have an infinite number of pages. There are then various objects in the pages that make up the page structure itself, which you can learn about below.

**Attention!**  Page 1 must always be called "page\_1"!

## Design demonstration in JSONu

The colors indicate the objects in the code below.

The base object in which the entire application is located.

The app page.

It is used for the design or function of the page. There are several types (more will be added gradually).

* **Window** - Defines the size, name, and other things for the application window.
* **Text** - Displays text in a specific defined location.
* **Button** - Pressing a button can do some action.
* **Text box** - Text box.
* **Border** - One-color block (picture can also be set)
* **Check box** - Check box.
* **Text box multiline** - Text box with multiple lines.
* **Web browser** - Web browser.

A detailed description of these parameters can be found in the next section.

{ “pages”: [  
  
 { ”page\_1”: [  
  
 { ”window”:  
 { ”title”: ”Welcome to PavlOS!”, ”app\_name”: ”PavlOS Introduction”, ”size\_x”: 850, ”size\_y”: 425, ”icon”: ”images/pavlos\_logo\_reworked\_transparent.png” }  
 },  
  
 { ”text”: [  
 { ”string”: ”Welcome to PavlOS!”, ”pos\_x”: 170, ”pos\_y”: 70, ”scale\_x”:true, ”scale\_y”:true, ”font\_size”: 40, ”color”: ”R=1, G=1, B=1, A=1” },  
 { ”string”: ”We’re glad that you’ve downloaded the latest version of PavlOS!”, ”pos\_x”: 30, ”pos\_y”: 170, ”scale\_x”:true, ”scale\_y”:true, ”font\_size”: 20, ”color”: ”R=0.15, G=0.15, B=0.15, A=1” }  
 ]  
 },  
  
 { ”button”: [  
 { ”string”: ”NEXT”, ”pos\_x”: 0, ”pos\_y”: 310, ”size\_x”: 850, ”size\_y”: 40, ”font\_size”: 20, ”resize\_x”:true, ”scale\_y”:true, ”text\_color”: ”R=0.08. G=0.08, B=0.08, A=1”, ”color”: ”R=0.7, G=0.7, B=0.7”, ”style”: ”styles/button/modern\_style\_dark.ini”, ”command”: [{”cmd”: ”open\_page: pages[1].what\_is”} ] }  
 ]  
 },  
  
 { ”text\_box”: [  
  
 ]  
 },  
  
 { ”border”: [  
 { ”pos\_x”: 0, ”pos\_y”: 50, ”size\_x”: 850, ”size\_y”: 100, ”zorder”: -1, ”resize\_x”:true, ”scale\_y”:true, ”color”: ”R=0.2, G=0.2, B=0.2, A=1” },  
 { ”pos\_x”: 35, ”pos\_y”: 50, ”size\_x”: 100, ”size\_y”: 100, ”zorder”: 1, ”scale\_x”:true, ”scale\_y”:true, ”color”: ”R=1, G=1, B=1, A=1”, ”img”: ”images/PavlOS\_logo\_transparent” },  
 ]  
 },  
 { ”check\_box”: [  
  
 ]  
 },  
  
 { ”text\_box\_multiline”: [  
  
 ]  
 },  
  
 { ”web\_browser”: [  
  
 ]  
 },  
 ]  
},  
  
 { ”what\_is”: [  
  
 { ”window”:  
 { ”title”: ”Welcome to PavlOS! – What is Pavl?”, ”app\_name”: ”PavlOS Introduction”, ”size\_x”: 850, ”size\_y”: 425, ”icon”: ”images/pavlos\_logo\_reworked\_transparent.png” }  
 },  
  
. . . . . . . . Continue the code with other objects . . . . . . . .  
  
 ]  
},  
 ]  
}

## Detailed description of objects in the page

Learn what objects can contain here. For example, window contains title, app\_name etc.

### Values

* Pink - String
* Green - Float
* Light Blue - Integer
* Red - Boolean
* Dark Blue - Vector (e.g. R=1, G=1, B=1, A=1)

### Window

* title - Window title.
* app\_name - Application name.
* icon - Application icon (icon path must be entered)
* size\_x - The size of the window on the X-axis.
* size\_y - Window size on the Y-axis.
* min\_size\_x - The smallest possible window size on the X-axis.
* min\_size\_y – The smallest possible window size on the Y-axis.
* allow\_nav\_btn - Enable navigation buttons.
* bg\_color - Sets the background color.

### Text

* string - Text (option to use "\n" for SHIFT+ENTER - skip to new line).
* pos\_x - Position on the X-axis.
* pos\_y - Position on the Y-axis.
* font\_size - Font size.
* scale\_x - Changes position according to the size of the window on the X-axis.
* scale\_y - Changes position according to the size of the window on the Y-axis.
* color - Sets the color of the text.

### Button

* string - Text in the button.
* style - The path to the config file style (e.g. "styles/button/modern\_style\_dark.ini"). More in the chapter "Styles".
* command - Array with objects with commands. More in the chapter "Commands in pushers".
* pos\_x - Position on the X-axis.
* pos\_y - Position on the Y-axis.
* size\_x - Size on the X-axis.
* size\_y - Size on na the Y-axis.
* font\_size - Font size.
* scale\_x - Changes position according to the size of the window on the X-axis.
* scale\_y - Changes position according to the size of the window on the Y-axis.
* resize\_x - Resizes according to the size of the window on the X-axis.
* resize\_y - Resizes according to the size of the window on the Y-axis.
* text\_ Color - Sets the color of the text.
* color - Sets the color of the button.

### Text box

* text - Text in a text box.
* text\_ID - ID for linking to Pavl+ and commands (e.g. getting text from box text).
* hint - Help in the text box (if the text box is blank, you will see help.
* config\_file - The path to the configuration file that we want to edit with the box (when using this value, the config\_type must be **true).**).
* section - The section we want to edit in the configuration file (when using this value must be config\_type with **true**value).
* key - The date key in the specified section (when using this value, the config\_type with true **true**value).
* pos\_x - Position on the X-axis.
* pos\_y - Position on the Y-axis.
* size\_x - Size on the X-axis.
* size\_y - Size on na the Y-axis.
* font\_size - Font size.
* scale\_x - Changes position according to the size of the window on the X-axis.
* scale\_y - Changes position according to the size of the window on the Y-axis.
* resize\_x - Resizes according to the size of the window on the X-axis.
* resize\_y - Resizes according to the size of the window on the Y-axis.
* config\_type - Whether the text box will be used to set up the configuration file .ini.

### Border

* img - If the path to the image is inserted, the image is displayed instead of the border.
* pos\_x - Position on the X-axis.
* pos\_y - Position on the Y-axis.
* size\_x - Size on the X-axis.
* size\_y - Size on na the Y-axis.
* zorder - Move to the foreground or background using layer numbers (-1 backward, 1 forward — you can use any numbers).
* scale\_x - Changes position according to the size of the window on the X-axis.
* scale\_y - Changes position according to the size of the window on the Y-axis.
* resize\_x - Resizes according to the size of the window on the X-axis.
* resize\_y - Resizes according to the size of the window on the Y-axis.
* color - Sets the color of the border/image.

### Check box

* text - Text in check box.
* config\_file - The path to the configuration file that we want to edit the text with the box.
* section - The section we want to edit in the configuration file.
* key - The date key in the specified section.
* pos\_x - Position on the X-axis.
* pos\_y - Position on the Y-axis.
* size\_x - Size on the X-axis.
* size\_y - Size on na the Y-axis.
* zorder - Move to the foreground or background using layer numbers (-1 backward, 1 forward — you can use any numbers).
* text\_ Color - Sets the color of the text.
* color - Sets the color of the check box.

### Text box multiline

* text - Text in a text box.
* text\_ID - ID for linking to Pavl+ and commands (e.g. getting text from box text).
* hint - Help in the text box (if the text box is blank, you will see help.
* config\_file - The path to the configuration file that we want to edit with the box (when using this value, the config\_type must be **true**).
* section - The section we want to edit in the configuration file (when using this value, the config\_type must be **true**).
* key - The date key in the specified section (when using this value, the config\_type with **true** value).
* pos\_x - Position on the X-axis.
* pos\_y - Position on the Y-axis.
* size\_x - Size on the X-axis.
* size\_y - Size on na the Y-axis.
* font\_size - Font size.
* scale\_x - Changes position according to the size of the window on the X-axis.
* scale\_y - Changes position according to the size of the window on the Y-axis.
* resize\_x - Resizes according to the size of the window on the X-axis.
* resize\_y - Resizes according to the size of the window on the Y-axis.
* config\_type - Whether the text box will be used to set up the configuration file .ini.

### Web browser

* url - The URL of the site you want to retrieve (Can be retrieved from the box text using its text\_ID - get. **vložte\_jeho\_text\_ID**).
* pos\_x - Position on the X-axis.
* pos\_y - Position on the Y-axis.
* size\_x - Size on the X-axis.
* size\_y - Size on na the Y-axis.
* zorder - Move to the foreground or background using layer numbers (-1 backward, 1 forward — you can use any numbers).
* scale\_x - Changes position according to the size of the window on the X-axis.
* scale\_y - Changes position according to the size of the window on the Y-axis.
* resize\_x - Resizes according to the size of the window on the X-axis.
* resize\_y - Resizes according to the size of the window on the Y-axis.

## Styles

Styles are used for custom designing (so far buttons) of objects – images when pressed, etc. Styles are written in the configuration file ".ini" and the path to this file is written to the object as "style".

### Create a configuration file for styles

The configuration file with the (modern) style here looks like this.

[normal] (object state)  
color=(R=0,G=0,B=0,A=0.075) (color)  
img=images/styles/white.png (image file path)  
[hovered]  
color=(R=0,G=0,B=0,A=0.1)  
img=images/styles/white.png  
[pressed]  
color=(R=0,G=0,B=0,A=0.05)  
img=images/styles/white.png

Before and after style button

## commands in pushers

The buttons are programmable at your will. There are a few commands you can use. Each of these commands is written to the "cmd" arraye in the "button" object in the JSON file.

run\_app: ( application in .json file with file path -e.g. apps/test.json or text\_ID from text box to get text)  
run\_app: test.json

close\_app (closes the app)

min\_app (minimizes application to bottom panel)

open\_page: (name of the app page you want to open)  
open\_page: page\_1

notification: (text notification)  
notification: The alert works!

shutdown\_OS (shuts down the operating system)

restart\_OS (restarts the operating system)

refresh\_settings (resets system settings – e.g. after changing system settings – is used in the "apply settings" button in the "settings" application)

play\_sound: (plays audio frome file .wav)  
play\_sound: sounds/system/shutdown.wav

trust\_app= (app name or"last app" – the system will trust this app – if the app contains unusual requirements, then your system will no longer notify you – unusual requirements are e.g. restart\_OS, refresh\_settings)  
trust\_app=last\_app (will believe the last opened application)  
trust\_app=apps/test.json (will trust the specified application)

@ (if the command contains "@" as the first character, then the compiler for pavl+code is automatically used - more in the programming chapter)  
@set=number1=5+number2  
@int=number3=546

# Programming

## Overview

It is programmed in our pavl+ language. The language is simple to understand and has a simple structure. More features will be added in future releases, so you can tell us what we can add. As in any language, there are variables, logical functions, etc.

Each function is written in one line — the code runs line by line.

With programming, you can create any application and have more room for creativity — for example, notifications can have text from a variable, or you can easily link code to JSON objects.

## Variable types

There are 4 types of variables – integer, float, string, boolean.

### Write variables

Variables are written in a similar way to other languages.

VARIABLE TYPE=VARIABLE NAME=VARIABLE VALUE

short entry:

Int=Number1=54  
Flt=Number2=8.56  
Str=Text1=Hello  
Bool=Bool1=True

normal entry:

Integer=Number1=54  
Float=Number2=8.56  
String=Text1=Hello  
Boolean=Bool1=True

### Use of variables

Variables are used to store data that your application needs to remember or work with.

## Library

### Set variable values

There are many ways to set the value of a variable. Mathematics works very similarly to, for example, c. So far there are a few mathematical tasks available, later more will be added (for example, it is not possible to add and multiply at the same time)

#### Set a value

SET=VARIABLE NAME=data (datais inserted into variable))

Set=string1=Hi mate! (INSERTTEXT INTO STRING)

Set=Number1=1568 (INSERTING A NUMBER INTO INTEGER)

Set=Number1=Number2 (INSERTING DATA INTO A VARIABLE FROM ANOTHER VARIABLE)

#### GET A VALUE FROM A JSON OBJECT

GET=variable name (data from object is inserted into this variable)=OBJECTTYPE=OBJECT ID (object id is set in object in json file)

Get=string1=text\_box=txtbox1

Get=number1=text\_box=txtbox1 (VLOŽENÍ čísla do integeru/floatu)

Get=bool1=check\_box=chbox1 vloží boolean do booleanu)

#### composed string of multiple variables or texts

append=variable name(variables and textsare inserted in this variable)=variables and texts (text is in quotation marks 'text')

Append=string1='Result: '+number1 (string in quotation marks 'text' and variable výwith number- result: 48)

Append=string1=string2+number1+string3 (string s číslem a dalším stringem)

#### Addition

SET=VARIABLE NAME(IN WHICH WE INSERT THE RESULT)=addition

Set=Number1=10+5 (adding numbers and inserting a vložení result into a variable)

Set=Number1=Number1+2 (adding a variable with a number and inserting a vložení result into a variable)

Set=Number1=Number1+Number2 (adding two variables and inserting a vložení result into a variable)

#### Subtraction

SET=VARIABLE NAME (IN WHICH WE INSERT THE RESULT)=SUBTRACTION

Set=Number1=10-5 (ODČÍTÁNÍ vložení

Set=Number1=Number1-2 (Subtraction of a variable with a number and insertion of the result into a variable)

Set=Number1=Number1-Number2 (SUBTRACTION of two variables and insertion of the result into a variable)

#### Multiplication

SET=VARIABLE NAME (TO WHICH WE INSERT THE RESULT)=MULTIPLICATION

Set=Number1=10\*5 ( MULTIPLYingnumbers and inserting the result into a variable)

Set=Number1=Number1\*2 ( MULTIPLYinga variable with a number and inserting a result into a variable)

Set=Number1=Number1\*Number2 (MULTIPLYing two variables and inserting a result into a variable)

#### Partition

SET=VARIABLE NAME (INTO WHICH WE INSERT THE RESULT)=division

Set=Number1=10-5 (dividing numbersand inserting the result into a variable)

Set=Number1=Number1-2 (dividingavariable with a number and inserting a vložení result into a variable)

Set=Number1=Number1-Number2 (dividing twovariables and inserting the result into a variable)

### Repeat (for loop)

For loop is used to repeat a particular function several times in a row (you specify how many times it will be repeated).

for=how many times it will be repeated (it can also be a variable)=@kód that will be performed ("@" always means - run code)

For=7=@set=number1=number1+1 (7x the command is repeated)

For=number2=@append=number1+'Hi' (the command is repeated as many times, according to the value of the variable "number2")

### Working with the system

There are a lot of things you can do or change in the system, such as restarting or shutting down the system, changing the date and time...

Sys=shutdown (vypne systém)

Sys=restart (restarts system)

Sys=close\_apps (closes all applications)

Sys=min\_apps (minimizes all applications)

Sys=notification=Title (Can be variable)=Text (Can be variable) (creates a notification immediately)

Sys=time\_and\_date=12 (sets 12-hour format with mm/dd/yyyy)

Sys=time\_and\_date=24 (sets 24-hour format with dd/mm/yyyy)

### Print to log

Writes the specified text to the system logo (not yet complete).

print=text or variable

Print=This works! (printne the sentence "this works!" to the log)

Print=number1 (printne into the log data from variable "number1")