JSL - Good Programming Skills

accantec

@accantec

p.bewerunge@accantec.com

Legacy Code is Fear

"any software you're afraid to change, but must change"
– Unknown–

Development of JSL Code in Team

- Coding Conventions
- Architecture Basics
- Testing Code

Coding Conventions: JSL

- encapsulate reusable Code in JSL functions
- One function = one file (.jsl)!
- Use function names that are easy to understand
- Function parameters start with "p_", continue camelCase. Comma to separate parameters
- JSL own "functions" in Uppercase
- Indent code within the function (1 x tab)
- Task:
  - Develop a function that performs a division of two values

```
division = FUNCTION({    p_denominator
                       ,    p_numerator
                        },
    result = p_numerator / p_denominator;
    RETURN(result);
);
```

Comments

"Code is comment enough..." yes ☐ no ☒ maybe ☐

- the more the better
- if possible, insert text from the concept
- Comment with: `/*comment*/`

```
/*-------------------------------------------------------*/
/*  That´s a really nice comment for the next steps      */
/*-------------------------------------------------------*/

/* And this is a pretty nice comment for the next step */

  Graph Builder (
  Variables( X( :weight ), Y( :height ) ),
  );
```

Development of JSL Code in a Team

Testing Code

Coding Conventions: Test JSL Code

- Insert at least one executable test at the end of a function. Best insert several tests with different parameters.
- Assign parameters with the same name. This makes it easier to " debug " the function. Sections within the code are testable.
- Use assigned function parameters when calling.
- Test boundary conditions (e.g. division by 0)
- At the end of the test, comment the test calls

```
division = function({    p_denominator
                       ,    p_numerator
                     },
    IF(p_denominator != 0,
        result = p_numerator / p_denominator;
        RETURN(result);
    ,//ELSE
        PRINT("Error: division by zero not allowed");
    );
);

/*Test-Area*/
p_denominator  = 0;
p_numerator    = 1;
c = division(   p_denominator    = p_denominator
              ,   p_numerator    = p_numerator
);
```

Architecture Basics

Architecture Basics

A house has always an architecture... a program too... the architecture can be good or bad...

- What happens if...
  - ... you take away a card?
  - ... you want to replace a card?
  - ... you want to add a new layer?
  - ... you want to document the architecture?

Quick & Dirty

"Nobody will remember the quick... but the dirty stays forever!”

Architecture basics

"Every piece of software has an architecture"

... modularized
... structured
... Reusable
... maintainable
... expandable
... documented
... tested

Architecture Basics: Developer Mantras

- *"I do create an architecture before I start"*
- *"I do think in "interfaces"... what does the function need as input, what does the program return (output)?"*
- *"I develop "small" modularized programs (no spaghetti code) "*
- *“I only save one JSL function per file (. jsl) -> same name"*
- *"I do apply the coding conventions "*
- *“I am allowed to adjust the architecture if necessary”*

Architecture Basics: How to develop an architecture?

- Transfer the chapters as a separate JSL function into a "wrapper program" (calculationBMI)

- Concept: BMI calculation
  1. data basis
     a) Data delivery from the previous system
     b) Interface Description
  2. data preprocessing
     a) filter data
  3. business logic
     a) Formula for calculating BMI
  4. Result export
     a) Export as Excel spreadsheet
  5. reporting
     a) Presentation of the results separated by gender
     b) Generate PDF report

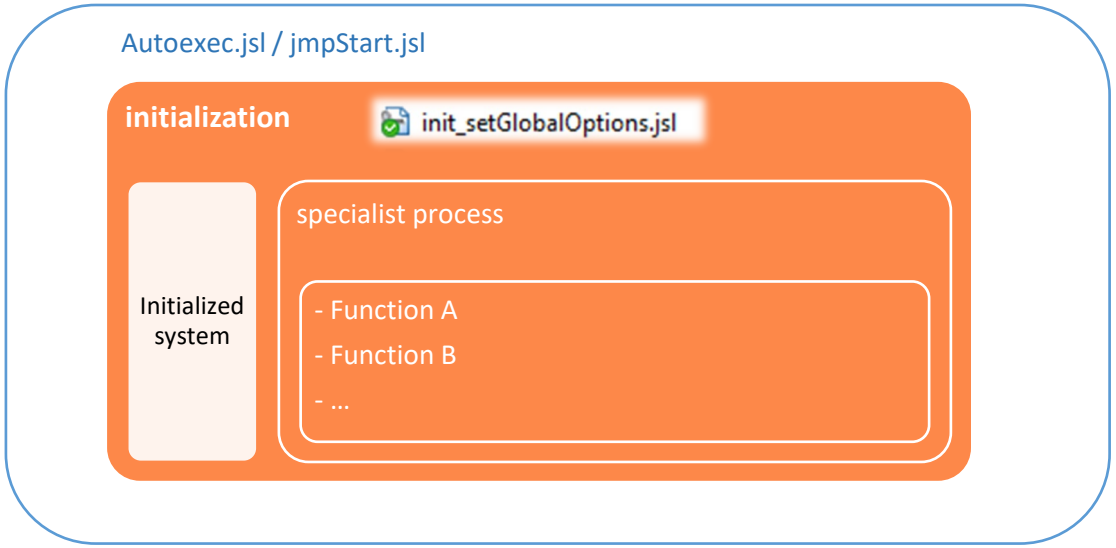
```
calculateBMI ({},
   importData ( {} );
   dataManipulation ( {} );
   computeBMI ( {} );
   exportResultsAsExcelTable ( {} );
   createReport ( {} );
);
```

Architecture Basics: Initialization

- The initialization program is the most important program of all:
  - it is the only place where global variables are defined and initialized
  - global variables can be used in functions , but never assign a value elsewhere
  - global variables start with "g_"
  - Typical content:
    - directory information
    - file names
    - Output library names
    - Product names (on which e.g. is filtered)

jmpStart.jsl
Runs every time JMP starts if...
1. C:\Users\<username>\AppData\Roaming\SAS\JMP\<version number >
2. C:\Users\<username>\AppData\Roaming\SAS\JMP
> Call for initialization
> Call of the specialist process

Or:
> Creation of your own menu entry



No Pain… No Gain… No Fingerprinted Code…

Dr. Peter Bewerunge

p.bewerunge@accantec.com

accantec group
Rudolf-Diesel-Str. 11
69115 Heidelberg
info@accantec.de
www.accantec.de