Bearskin Hats

Using Guard Clauses for More Robust Programming

February 8, 1:30 pm–2:30pm

Cristos Lianides-Chin

codence

Who Am I?

Platform Architect at Codence

Worked in FileMaker since 2012 (*no `.fp5` conversions*)

Passionate about cooking, sailing, and ~~lazy programming~~ **efficient coding practices**

cristos.lianides-chin@codence.com

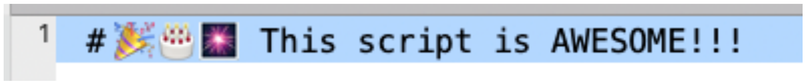
codence

Take Aways

01

New Developers

- You can use emojis in FileMaker script comments



02

Intermediate Developers

- Build a foundational understanding of how to use **guard clauses**
- Learn how to implement the **Try-Catch** pattern in FileMaker.

03

Tech Leads

- Learn to apply guard clauses more effectively, especially in complex scripting scenarios.
- Gain broader knowledge on coding best practices.

What We'll Cover

✅ In Scope

Guard Clauses

Single-Pass Loops / Try-Catch

❌ NOT In Scope

Error Handling

- Robert Naud | Error Handling Revisited *(DIGFM 2023-10, https://youtu.be/muhK5AGVb7o)*

Transactions

- Alondra Torres-Navarro & Barbara Cooney | Advanced modern Claris FileMaker transactions *(ENGAGE 2024)*

Guard

```
$var = Get ( ScriptParameter )
...
```

```
...
# Do some more stuff
Perform Script [ “Profit!” ; $var ]
Exit Script []
```

(Demo file)

Guard

$var = Get (ScriptParameter)

...

💂 If [requirementsNotMet]:

Exit Script []

...

~~# Do some more stuff~~

~~Perform Script ["Profit!" ; $var]~~

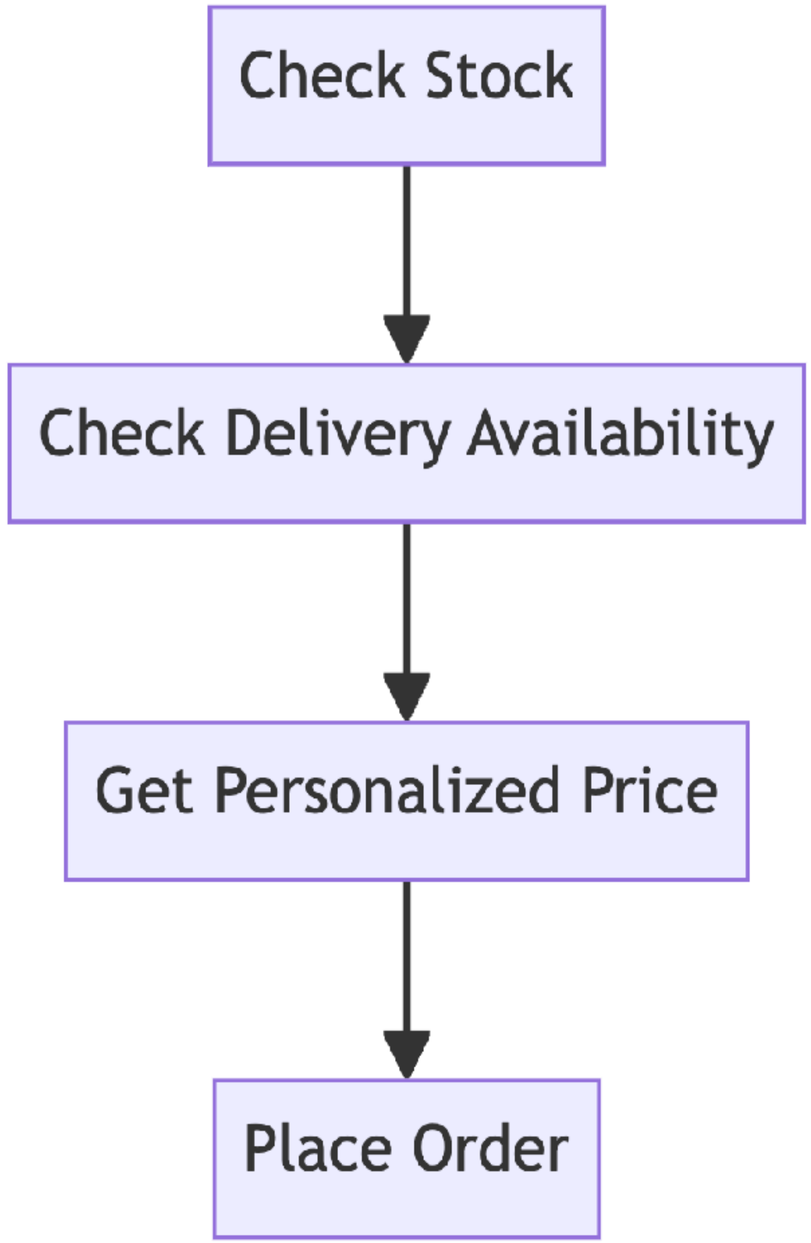
(Demo file)

[DEMO]
If vs Guard

1 Customer
+ 1 Product
= 1 Order

(Demo file)

[DEMO] Order Process



(Demo file)

Why Guard Clauses?

01

More Readable Code

Avoid the Arrow Pattern

02

Fail Fast

Don’t perform steps that you know will fail or return garbage data.

03

Further Reading

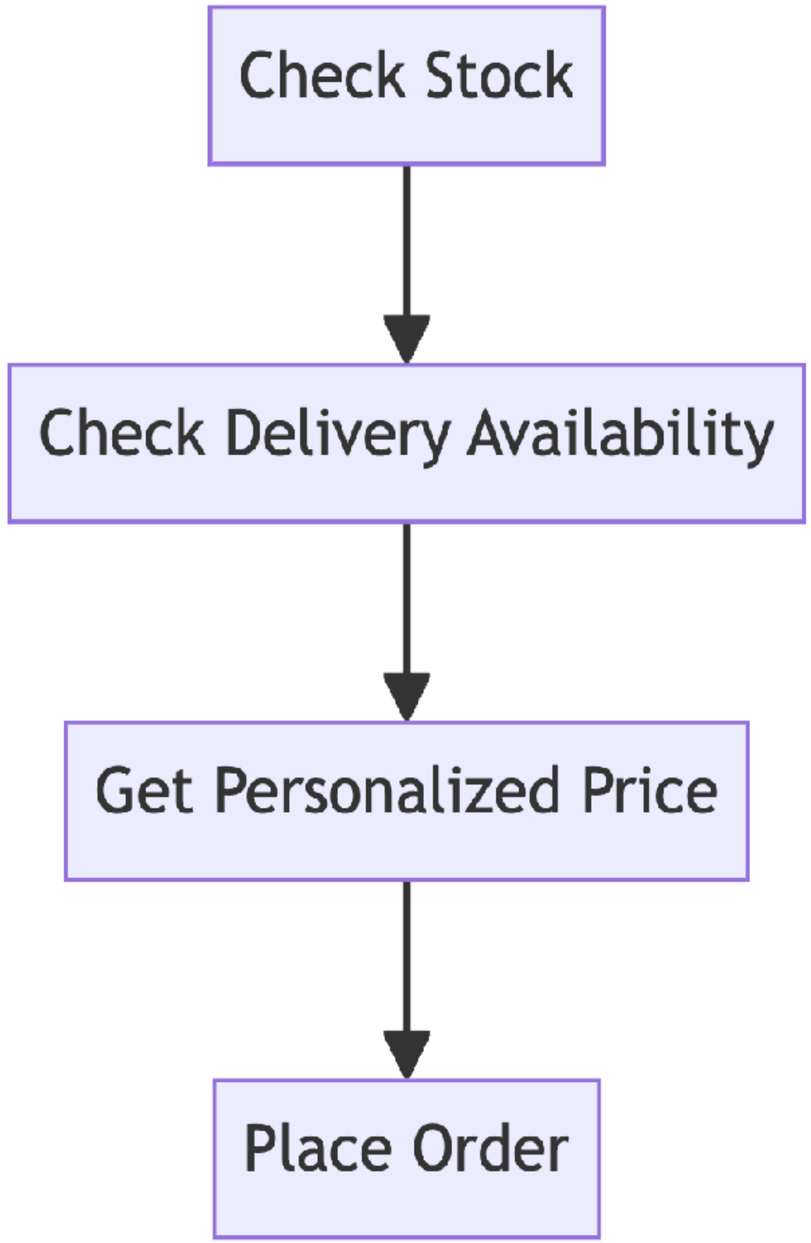
- https://www.codementor.io/@clintwinter/use-guard-clauses-for-cleaner-code-1rrsczgwxp
- https://en.wikipedia.org/wiki/Guard_(computer_science)

(Demo file)

"It is not enough for code to work. It must also be clean, readable, and understandable."

~ Robert C. Martin, *Clean Code*

[DEMO]
Exit Script
v Exit Loop If



(Demo file)

Try-Catch

Especially Useful in Iterators

```
data = ["10", "20", "a", "30", "40b", "50"]

for item in data:
    try:
        number = int(item)  # Attempt to convert string to integer
        print(f"Processed number: {number}”)

    except ValueError:
        print(f"Invalid item encountered: '{item}' - Skipping")
```

```
Processed number: 10
Processed number: 20
Invalid item encountered: 'a' - Skipping
Processed number: 30
Invalid item encountered: '40b' - Skipping
Processed number: 50
```

(Demo file)

Try-Catch in FileMaker

Single Pass Loop

- Doesn’t affect transaction
- Can see error message in Script Workspace
- Can take practice to scan for error message

(Demo file)

TRY

```
Loop
   # We haven't placed the order yet, so set the initial value
   Set Variable [ $isOrderPlaced ; Value: False ]

   # 🥊 Check if we have required parameters
   Exit Loop If [ If ( IsEmpty ( $idRecord ) ; Let ( $error = "missing $idRecord" ; True
   Exit Loop If [ If ( IsEmpty ( $idCustomer ) ;  Let ( $error = "missing $idCustomer" ;

   Set Variable [ $objOrderRequested ;
   Value: JSONSetElement ( "{}" ;  [ "idCustomer" ; $idCustomer ; JSONString ] ;[ "idProd

   # Open a new window because the remaining subscripts may change our context
   Set Variable [ $windowName ; Value: Get ( ScriptName ) & "__" & Get ( UUID ) ]
   New Window [ Style: Document ; Name: $windowName ; Using layout: <Current Layout> ]
   # ☆🥊 Are we in the new window now?
   Exit Loop If [ If ( not ( Get ( WindowName ) = $windowName ) ;Let ( $error = "" ; True
   Exit Script [ Text Result: JSONSetElement ( "{}" ; [ "isOrderPlaced" ; $isOrderPlaced ;

   Perform Script [ Specified: From list ; “Check Stock ( idProduct ) > qtyInStock, qtyOnBa
   Set Variable [ $qtyInStock ; Value: JSONGetElement ( Get ( ScriptResult ) ; "qtyInStock"

   # 🥊 Check if the ordered item is in stock.
   Exit Loop If [ If ( 0 = $qtyInStock ; Let ( $error = "Insufficient stock on hand." ; T

   Perform Script [ Specified: From list ; “Check Delivery Availability ( objOrderReqeusted
   $objOrderRequested ]
   Set Variable [ $isDeliverable ; Value: JSONGetElement ( Get ( ScriptResult ) ;"isDeliver

   Exit Loop If [ True // END TRY ]
End Loop
```

CATCH

```
If [ Length ( $error ) ]
   # e.g., Log the Error
End If
```

FINALLY

```
# FINALLY
Close Window [ Name: $windowName ; Current file ]

Set Variable [ $scriptResult ;
Value: JSONSetElement ( "{}" ;  [ "isOrderPlaced" ; $isOrderPlaced ; JSONBoolean ] ;[ "e
Exit Script [ Text Result: $scriptResult ]
```

Try-Catch in FileMaker

Open / Commit Transaction

- More bang for your buck
- May not fit all situations
- Can’t see error message without clicking through

(Demo file)

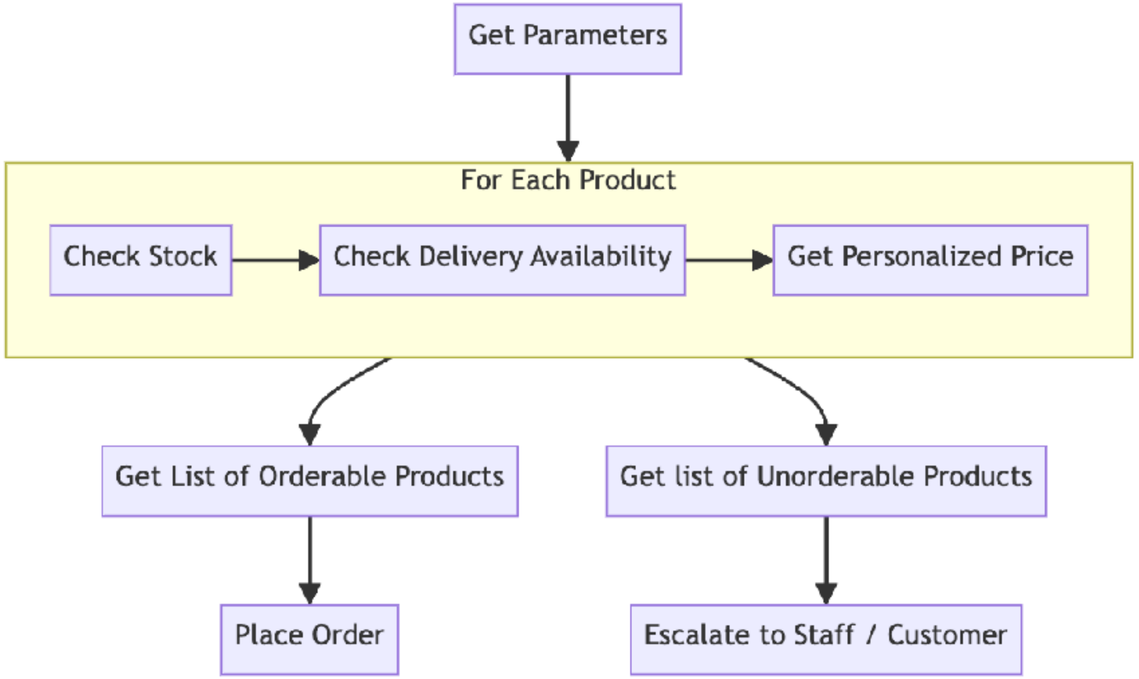
TRY

CATCH

FINALLY

```
21  Open Transaction [ ]
22     # We haven't placed the order yet, so set the initial value
23     Set Variable [ $isOrderPlaced ; Value: False ]
24
25     # 🥊 Check if we have required parameters
26     Revert Transaction [ Condition: IsEmpty ( $idCustomer ) ]
27     Revert Transaction [ Condition: IsEmpty ( $idProduct ) ]
28
29     Set Variable [ $objOrderRequested ;
       Value: JSONSetElement ( "{}" ;  [ "idCustomer" ; $idCustomer ; JSONString ] ;[ "idP
30
31     # Open a new window because the remaining subscripts may change our context
32     Set Variable [ $windowName ; Value: Get ( ScriptName ) & "__" & Get ( UUID ) ]
33     New Window [ Style: Document ; Name: $windowName ; Using layout: <Current Layout> ]
34     # ⭐🥊 Are we in the new window now?
35     Revert Transaction [ Condition:  not ( Get ( WindowName ) = $windowName ) ]
36
37     Perform Script [ Specified: From list ; “Check Stock ( idProduct ) > qtyInStock, qtyOn
38     Set Variable [ $qtyInStock ; Value: JSONGetElement ( Get ( ScriptResult ) ; "qtyInSto
39
40     # 🥊 Check if the ordered item is in stock.
41     Revert Transaction [ Condition: 0 = $qtyInStock ]
42
43     Perform Script [ Specified: From list ; “Check Delivery Availability ( objOrderReqeust
       $objOrderRequested ]
44     Set Variable [ $isDeliverable ; Value: JSONGetElement ( Get ( ScriptResult ) ;"isDeli
66  Commit Transaction
67
68  If [ Length ( $error ) ]
69     # e.g., Log the Error
70  End If
71
72  # FINALLY
73  Close Window [ Name: $windowName ; Current file ]
74
75  Set Variable [ $scriptResult ;
    Value: JSONSetElement ( "{}" ;  [ "isOrderPlaced" ; $isOrderPlaced ; JSONBoolean ] ;[
76  Exit Script [ Text Result: $scriptResult ]
77
78
```

[DEMO] Try-Catch in Other Structures



(Demo file)

Scripts vs Calcs

Calc engine doesn't have "Try-Catch"

```
Case
(   // Fastest checks go towards the top
    IsEmpty ( $parameter )   // 🥊
    ;   "? - Error: parameter is empty"

;   Left ( JSONFormatElements ( $parameter ) ; 1 ) = "?"    // 🥊
;       "? - Error: parameter is not valid JSON"

;   Let ( [ ~md5 = GetContainerAttribute ( $parameter ; "Md5" )]
        ; PatternCount ( ~md5 ; $listOtherMd5Hashes ) > 0   // 🥊
        )
;       "? - Error: parameter occurs multiple times in the set"


    // Slower checks go at the bottom
;   not Exact ( $parameter ; "TABLE::Field" )   // 🥊
;       "? - Error: parameter does not match database value"

;   Exact ( $parameter ; "TABLE::Field" )       // 🥊
    and Exact ( $parameter ; "RelatedTable::Field" )


;   // ELSE (main logic)
        Let
        (   // Load data for calc here
        [   ~fieldData = "TABLE::Field"
        ;   ~relatedFieldData = "RelatedTable::Field"
        ]
        ;   $parameter & ~fieldData & ~relatedFieldData
        )
)
```

Scripts vs Calcs

Calc engine doesn't have "Try-Catch"

...but you can still return early

```
Case
(   // Fastest checks go towards the top
    IsEmpty ( $parameter )   // 🥊
    ;   "? - Error: parameter is empty"

;   Left ( JSONFormatElements ( $parameter ) ; 1 ) = "?"    // 🥊
;       "? - Error: parameter is not valid JSON"

;   Let ( [ ~md5 = GetContainerAttribute ( $parameter ; "Md5" )]
        ; PatternCount ( ~md5 ; $listOtherMd5Hashes ) > 0   // 🥊
        )
;       "? - Error: parameter occurs multiple times in the set"


    // Slower checks go at the bottom
;   not Exact ( $parameter ; "TABLE::Field" )   // 🥊
;       "? - Error: parameter does not match database value"

;   Exact ( $parameter ; "TABLE::Field" )       // 🥊
    and Exact ( $parameter ; "RelatedTable::Field" )


;   // ELSE (main logic)
        Let
        (   // Load data for calc here
        [   ~fieldData = "TABLE::Field"
        ;   ~relatedFieldData = "RelatedTable::Field"
        ]
        ;   $parameter & ~fieldData & ~relatedFieldData
        )
)
```

Scripts vs Calcs

Calc engine doesn't have "Try-Catch"

...but you can still return early

```
Case
(   // Fastest checks go towards the top
    IsEmpty ( $parameter )   // 🍷
    ;   "? - Error: parameter is empty"

;   Left ( JSONFormatElements ( $parameter ) ; 1 ) = "?"    // 🍷
;       "? - Error: parameter is not valid JSON"

;   Let ( [ ~md5 = GetContainerAttribute ( $parameter ; "Md5" )]
        ; PatternCount ( ~md5 ; $listOtherMd5Hashes ) > 0   // 🍷
        )
;       "? - Error: parameter occurs multiple times in the set"


    // Slower checks go at the bottom
;   not Exact ( $parameter ; "TABLE::Field" )   // 🍷
;       "? - Error: parameter does not match database value"

;   Exact ( $parameter ; "TABLE::Field" )       // 🍷
    and Exact ( $parameter ; "RelatedTable::Field" )


;   // ELSE (main logic)
        Let
        (   // Load data for calc here
        [   ~fieldData = "TABLE::Field"
        ;   ~relatedFieldData = "RelatedTable::Field"
        ]
        ;   $parameter & ~fieldData & ~relatedFieldData
        )
)
```

Scripts vs Calcs

Calc engine doesn't have "Try-Catch"

...but you can still return early

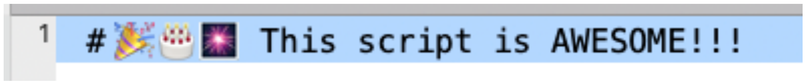
```
Case
(   // Fastest checks go towards the top
    IsEmpty ( $parameter )   // 🥊
    ;   "? - Error: parameter is empty"

;   Left ( JSONFormatElements ( $parameter ) ; 1 ) = "?"    // 🥊
;       "? - Error: parameter is not valid JSON"

;   Let ( [ ~md5 = GetContainerAttribute ( $parameter ; "Md5" )]
        ; PatternCount ( ~md5 ; $listOtherMd5Hashes ) > 0   // 🥊
        )
;       "? - Error: parameter occurs multiple times in the set"


    // Slower checks go at the bottom
;   not Exact ( $parameter ; "TABLE::Field" )   // 🥊
;       "? - Error: parameter does not match database value"

;   Exact ( $parameter ; "TABLE::Field" )       // 🥊
    and Exact ( $parameter ; "RelatedTable::Field" )


;   // ELSE (main logic)
        Let
        (   // Load data for calc here
        [   ~fieldData = "TABLE::Field"
        ;   ~relatedFieldData = "RelatedTable::Field"
        ]
        ;   $parameter & ~fieldData & ~relatedFieldData
        )
)
```

Take Aways

01

New Developers

- You can use emojis in FileMaker script comments



02

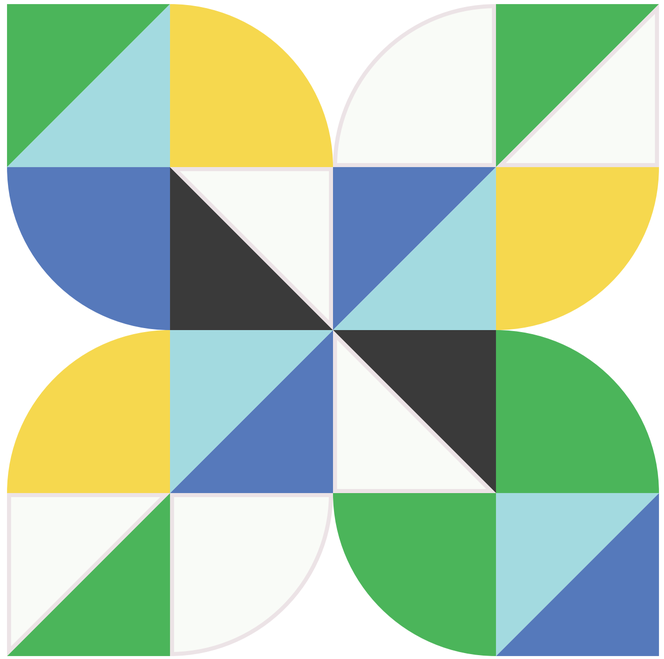
Intermediate Developers

- Build a foundational understanding of how to use **guard clauses**
- Learn how to implement the **Try-Catch** pattern in FileMaker.

03

Tech Leads

- Learn to apply guard clauses more effectively, especially in complex scripting scenarios.
- Gain broader knowledge on coding best practices.



Q&A

How to Type Emoji Comments

(Demo file)

Thank you!

cristos.lianides-chin@codence.com

https://linkedin.com/in/cristoslc