Publications Office
of the European Union

MOVING TO SDK-2

Ioannis Rousochatzakis – TED-together 2025

We will talk about…

- what is in SDK-2,
- our progress,
- and what to expect

Before we start…

Software Development Kit (SDK)

- The SDK is a message from the Publications Office to the eSender network
- Contains every detail of what eForms is and how to work with it

Without the SDK communication of the eForms specification would be slow and ambiguous.

eForms Expression Language (EFX)

- EFX is a convention that allows us to express business logic and rules.
- It is specific to eForms, and unambiguous.

Without EFX, we would be unable to communicate business logic and rules over the eSender network.

Strategic choices for SDK-2

Advance EFX as much as possible;

Bring everyone onboard as soon as possible.

Why?

EFX enables and enforces:

- Our shared understanding of all eForms components and their interactions
- Our ability to share business logic & rules across our data collection network

Tactical choice

Remain on SDK-2 long enough, but make room for incremental improvements.

Why?

Some eSenders can innovate faster than others. Helping all eSenders maintain their own pace:

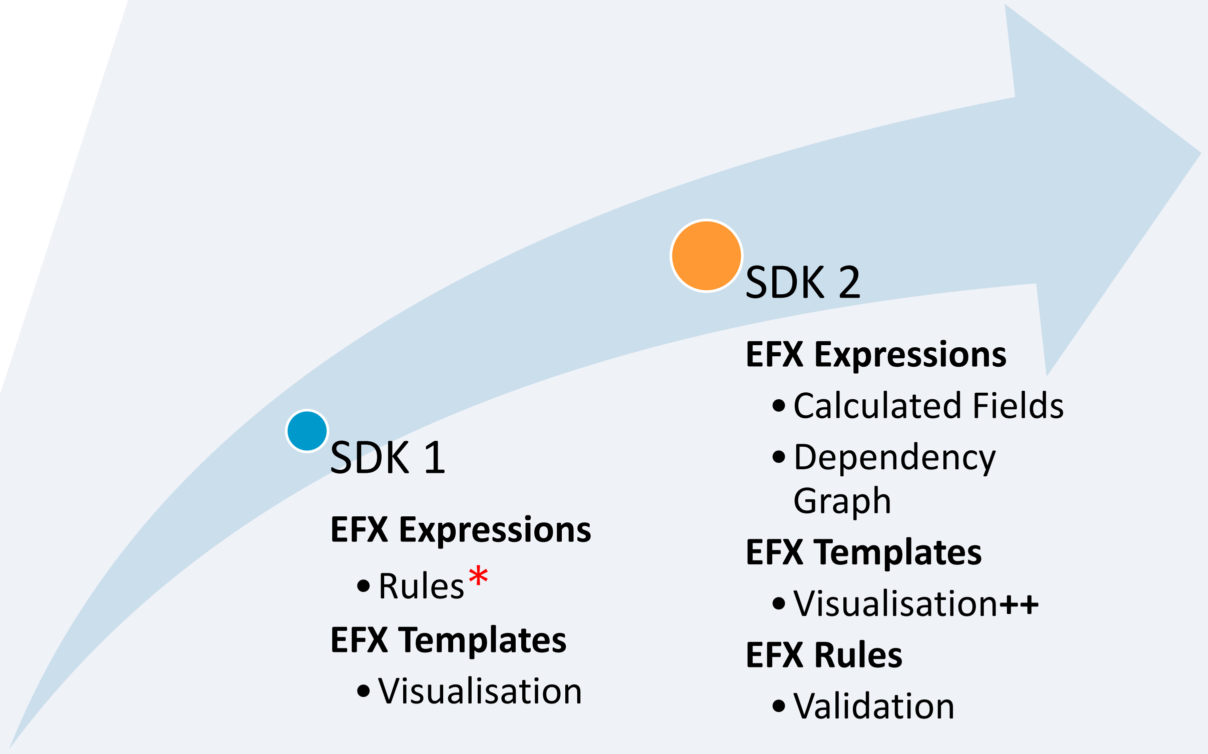
- will benefit the entire network
- will increase our collective preparedness for technological and legislative changes to come

Publications Office
of the European Union

IMPROVING EFX

Quick EFX recap

- Domain-specific language for eForms
- Powers eForms business logic:
  - computation,
  - validation,
  - visualisation



SDK 1

EFX Expressions
- Rules*

EFX Templates
- Visualisation

SDK 2

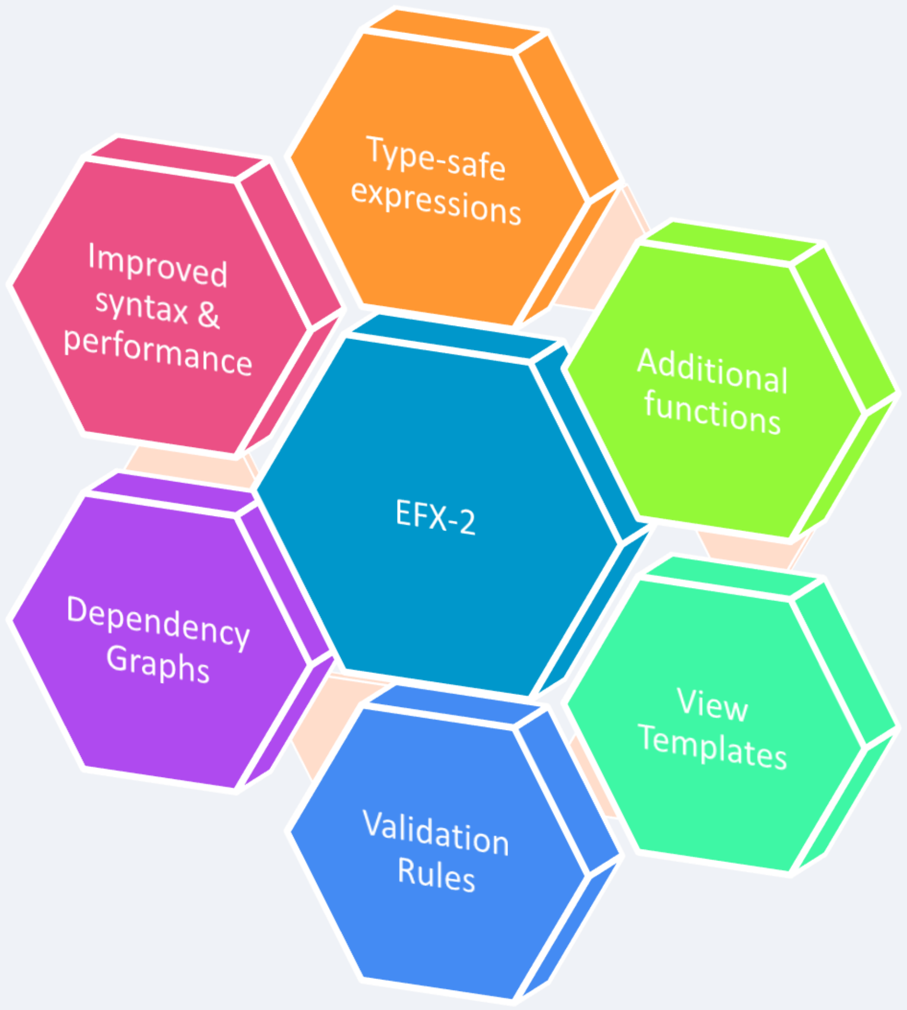
EFX Expressions
- Calculated Fields
- Dependency Graph

EFX Templates
- Visualisation++

EFX Rules
- Validation

EFX-2

- Improves notice visualisation
  - Adds missing features
  - Improves rendering performance
- Improves overall syntax
  - More readable
  - More advanced features
- Adds EFX Rules
  - Simpler and stricter definitions
  - Better suited for validation beyond Schematron



Publications Office
of the European Union

CHAPTER I – EFX RULES

A new EFX language flavour for validation

slido

EFX Rules - opening questions

What are we trying to achieve?

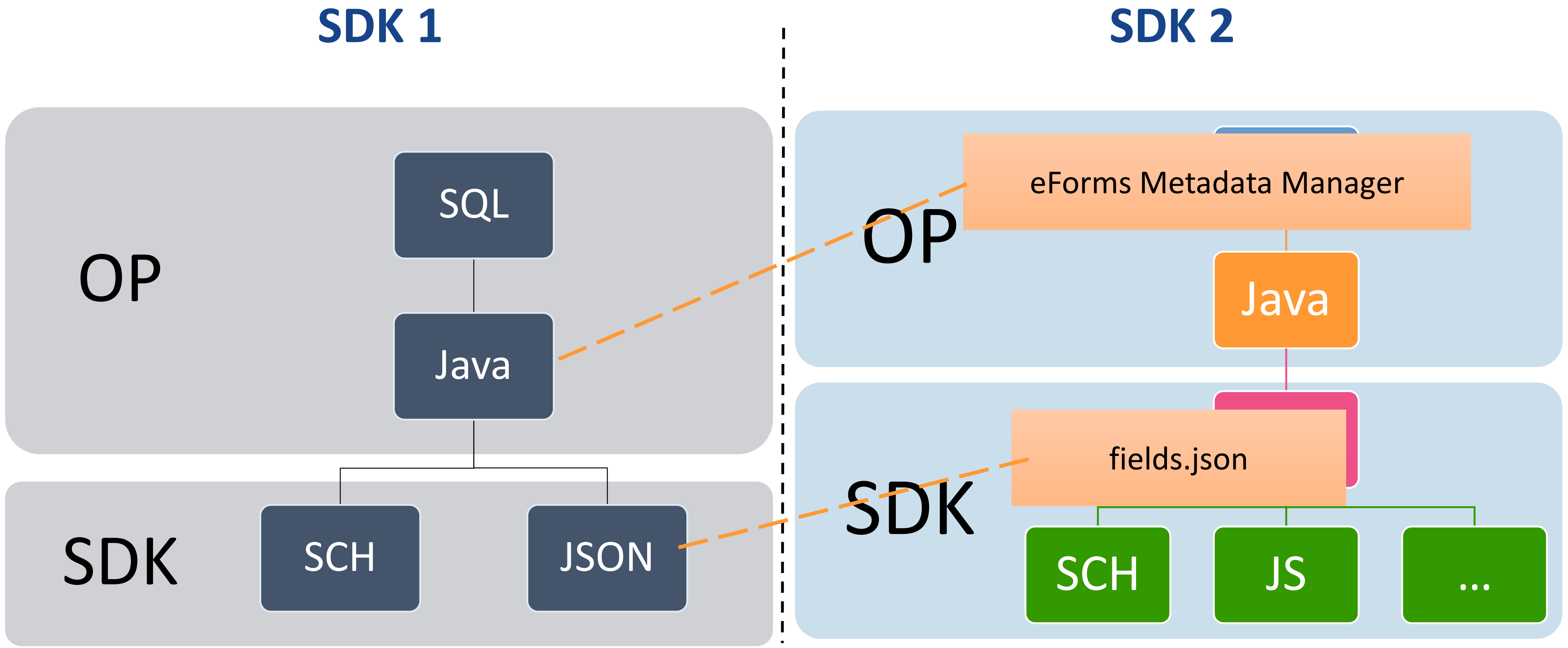
We realise that…

- We designed with Schematron in mind
  - Rules assume the entire notice has been filled in.
  - EFX is only used only for computation.
  - Rule definitions are still loose.
- Client-side validation has additional needs
  - Some rules should not run client-side.
  - Some rules cannot run without all fields needed for computation.
  - Rule dependencies can be deep and even circular.

We sought to improve…

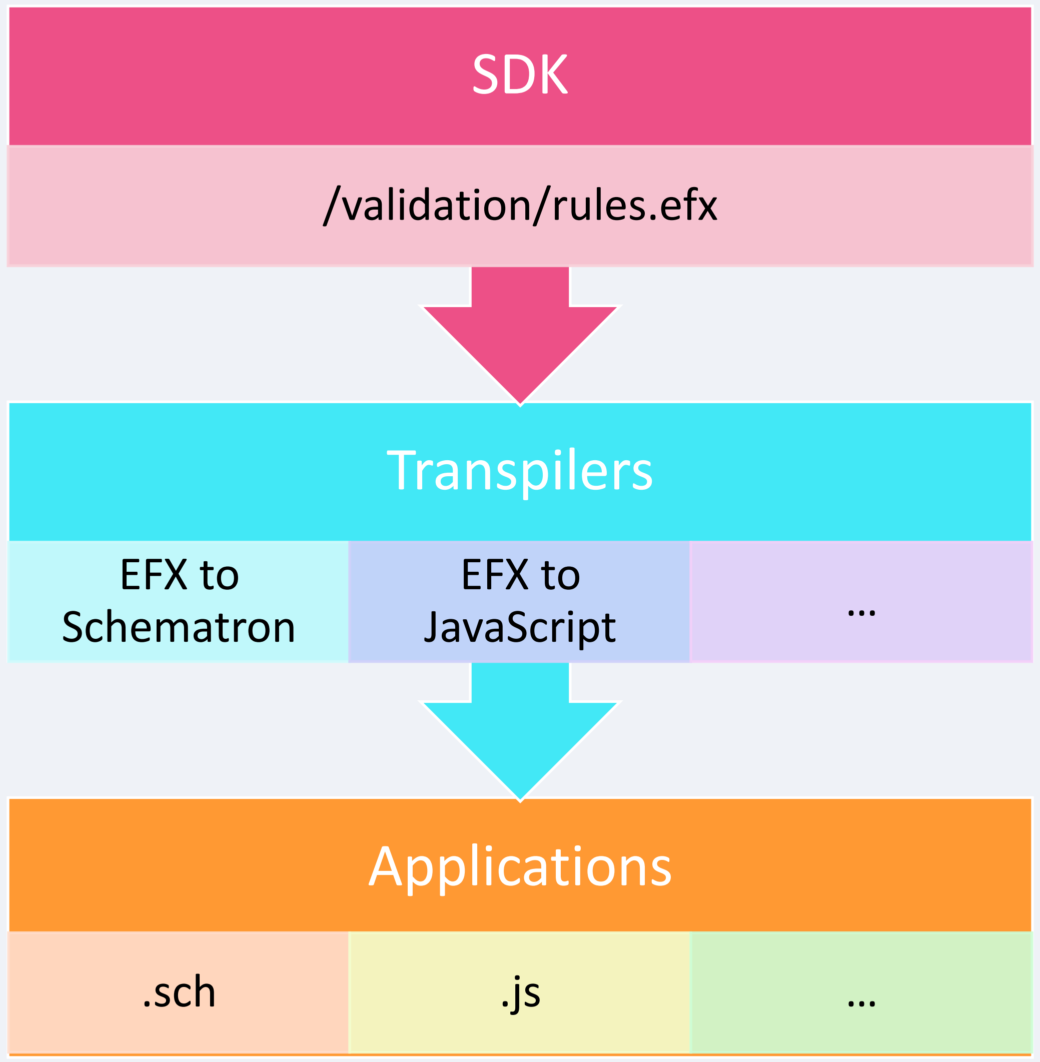
- Reduce repetition
  - Share rules across notice subtypes
- Further formalise rule definitions
  - Use EFX to standardise rule statements
- Target multiple validation engines through EFX
  - Keep Schematron as a target
  - Add JavaScript validation for web apps
  - Enable eSenders to target their preferred environments

Business Rules in the SDK



All rules in one EFX file

- `/validation` folder
  - `rules.efx`
- Transpilers (EFX Toolkit, etc.)
  - EFX to Schematron
  - EFX to JavaScript
  - EFX to *<any>*
- Post-validation (XML)
  - Validating entire notices
  - Schematron
- Client-side validation
  - Validation while filling the form
  - JS, etc.



SDK-1 – fields.json

```
 "assert" : {
      "value" : "{ND-Root} ${TRUE}",
      "severity" : "ERROR",
      "constraints" : [ {
        "condition" : "{ND-Root} ${(BT-01-notice == 'other') and (OPP-070-notice in ('1','2','3','4','5','6','7','8','9','10','11','12','13','14','15','16','17','18','19','20','21','22','23','24','25','26','27','28','29','30','31','32','33','34','35','36','37','38','39','40','E1','E2','E3','E4','E5','E6'))}",
        "value" : "{ND-Root} ${(BT-01(c)-Procedure is present) or (BT-01(f)-Procedure is present)}",
        "severity" : "ERROR",
        "message" : "rule|text|BR-BT-00001-0259"
      } ]
    }
```

SDK-2 – same rule in EFX-2

```
WITH ND-Root
    WHEN (BT-01-notice == 'other’)
    ASSERT (BT-01(c)-Procedure is present) or (BT-01(f)-Procedure is present)
     AS ERROR R-B3D-DF9
     FOR BT-01-notice
     IN 1-40, E1-E6;
```

Rule syntax in EFX-2

```
WITH <context>
   WHEN <condition>
   ASSERT <test>
    AS <SEVERITY> <rule-id>
    FOR <field-or-node>
    IN <notice-subtypes>,
  WHEN <condition>
  ASSERT <test>
    AS <SEVERITY> <rule-id>
    FOR <field-or-node>
    IN <notice-subtypes>;
```

/validation/rules.efx

```
--- STAGE 3b ---

WITH <context1>
  WHEN ... ASSERT ... FOR ... IN ...,
  WHEN ... ASSERT ... FOR ... IN ...;
WITH <context2>
  ASSERT ... FOR ... IN ...,
  WHEN ... ASSERT ... FOR ... IN ...;


--- STAGE 4 ---

WITH ...
```

Dependency graph

- Essential for client-side validation
- Two types of dependency
  - "**compute**" dependencies:
    Which other fields does a calculated field need?
  - "**assert**" dependencies:
    Which other fields are needed to validate a given field?
- Forward & reverse lookup
  - `dependsOn`
  - `requiredBy`
- Example: User changes the value of a field
  - Q: Which calculated fields need to be recalculated?
    A: `requiredBy.compute`
  - Q: Do we have the values of all other fields needed to validate the changed value?
    A: `dependsOn.assert`

```
{
  "fields": [
    {
      "id": "BT-105-Procedure",
      "dependsOn": {
        "compute": {
          "fields": [],
          "nodes": [],
          "codeLists": []
        },
        "assert": {
          "fields": ["BT-106-Procedure", "BT-01-notice"],
          "nodes": ["ND-ProcedureTenderingProcess"],
          "codeLists": []
        }
      },
      "requiredBy": {
        "compute": {
          "fields": [],
          "nodes": []
        },
        "assert": {
          "fields": ["BT-52-Lot", "BT-644-Lot", "BT-661-Lot",
                     "BT-88-Procedure", "BT-98-Lot"],
          "nodes": []
        }
      }
    },
    ...
```

Things to know

- Rules are “compressed” to around 3.000 rules
  - In SDK-1 we have approximately 40.000 rules
- As a result rule labels will also be reduced.
- Rule identifiers must change
  - Not set in stone yet
  - Randomised - No semantics
  - Considering **`R-XXX-XXX`**
    where X is a letter or digit (A-Z, 0-9) randomly generated

Please download and install the Slido app on all computers you use

What's your first reaction to EFX Rules?

ⓘ Start presenting to display the poll results on this slide.

Publications Office of the European Union

CHAPTER II – EFX TEMPLATES

What are we trying to achieve?

We struggled with…

- Simplistic templating model
  - EFX-1 focused on “what to display where” but missed the big picture of creating a fast, friendly, navigable document
- Repetitive expensive calculations
  - Lack of variables forced us to create long, expensive, repetitive calculations.
- Limited set of built-in functions
  - A richer set of built-in functions would also have helped to write simpler better performing expressions.
- Hard to read expressions

Therefore we worked on…

- Introducing variables with controlled scope
  - To reduce repetitive computation
- User defined functions & reusable templates
  - To simplify code and reduce code repetition
- Dictionaries
  - For faster key/value lookups
- Conditional templates
  - Reduce complexity and improve clarity
- Richer set of built-in functions
- Clearer syntax

New template syntax

```
WITH <context> DISPLAY <template>;
// Repeats the template for each matching context

WITH <context>
     WHEN <condition1> DISPLAY <template1>
     WHEN <condition2> DISPLAY <template2>
     OTHERWISE         DISPLAY <template3>;
// Repeats with context, but selects appropriate template based on conditions each time
```

Global variables, functions & reusable templates

```
LET text:$myVariable = ‘value’;
// declares a global variable

LET number:?myFunction(number:$parameter) = $parameter + 1;
// defines a custom function

LET template:myTemplate(text:$parameter) DISPLAY template, $parameter;
// declares a callable template

LET $myDictionary INDEX OPT-302-Organization BY BT-501-Organization-Company;
// creates a key/value dictionary for fast lookups

// usage
WITH ...
   WHEN ?myFunction(BT-113-Lot) > 10     // calls a function
   INVOIKE myTemplate(BT-01-notice);     // calls a template
```

Context variables

- Avoid repetitive calculations
- Scoped within nested template blocks

```
WITH ND-Organization, text:$orgId = OPT-200-Organization-Company DISPLAY ...;
   WITH OPT-200-Organization-Company[OPT-300-Procedure-Buyer == $orgid] DISPLAY ...;
```

Other improvements

- New “Summary” section provides controlled summary view.
- New “Navigation“ section provides controlled navigation over large notices.
- Intrinsic language selection for multilingual fields
- Intrinsic handling of privacy controls
- Hyperlinks

```
// full notice view template
WITH ... DISPLAY ... ;
   WITH ...
   ...

--- SUMMARY ---
// shorter summary view template
WITH ...
...

--- NAVIGATION ---
// navigational hierarchy
...
```

Legacy syntax (EFX-1 templates)

- EFX-2 supports view templates written in EFX-1
- Should we remove it?
  - Only if it will not disrupt you
  - Or if you all agree to remove it
- Why it would be good to remove it?
  - Unnecessary complexity in the language
  - Degrades performance significantly
  - Harder to maintain
- Better to drop EFX-1 syntax if not needed anymore

slido

EFX Templates - closing questions

Publications Office
of the European Union

CHAPTER III - EFX EXPRESSIONS

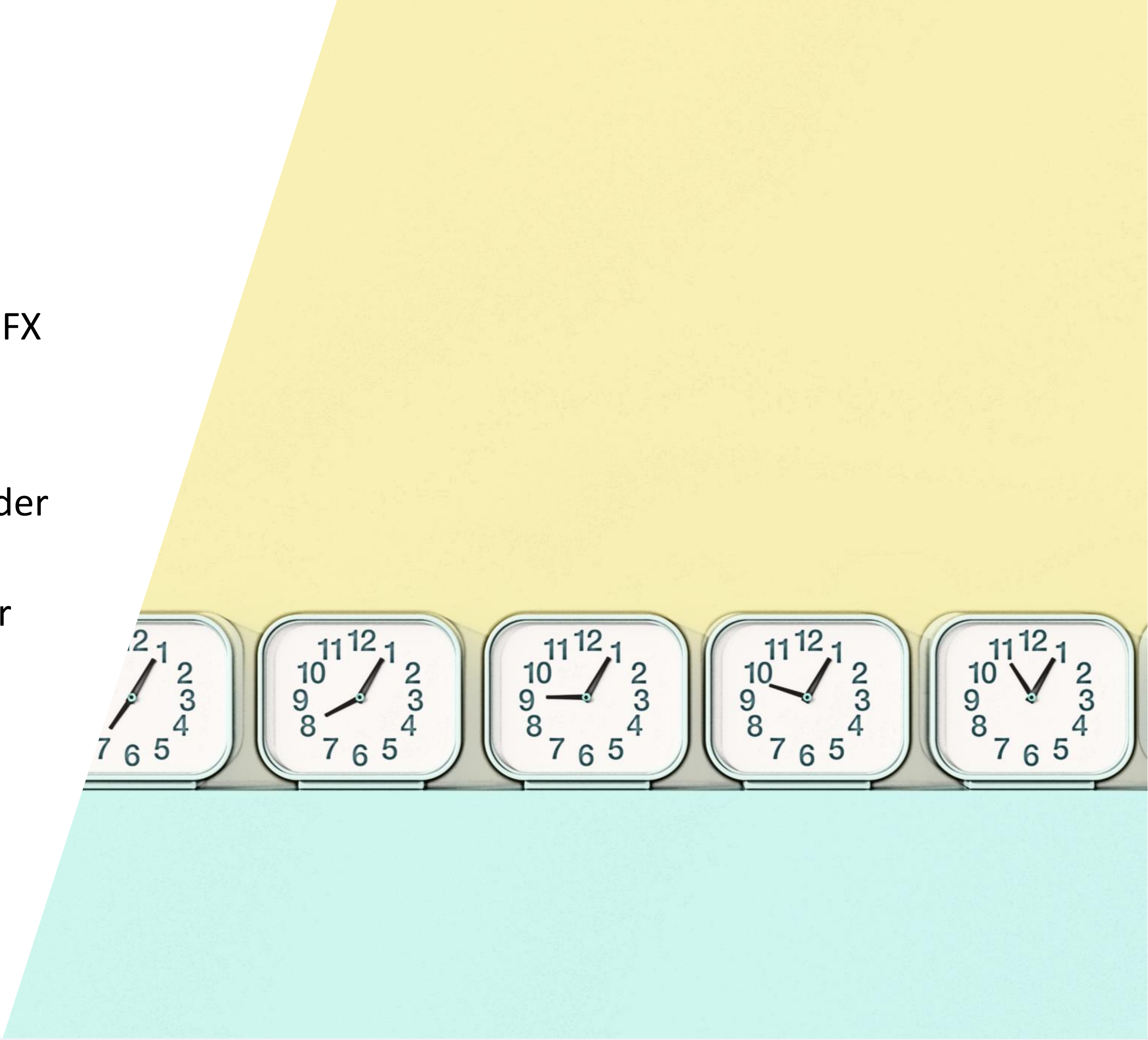
The foundation of the EFX language

EFX-2 expressions in a nutshell

- Foundation of EFX
- Remains strongly typed.
- A pre-processor is used to determine types of late-bound expressions.
- Better differentiates between scalar values and sequencies.
- Mostly backward compatible; two exceptions:
  - Type casting now uses parenthesis → `(type-name) identifier`
  - Codelist references now use a # prefix → `#codelist-name`
- Adds new functions to better process strings, dates, numbers, sequences
- Will also be used to define calculated fields

Timeline

- Upcoming alpha versions of SDK-2 and EFX Toolkit for Java within 2025
- Beta versions expected in February
- More definitive timeline by March eSender Workshop.
- Release as soon as EFX-2 is ready – other improvements to follow



Publications Office
of the European Union

FORWARD COMPATIBILITY

Staying ahead of the game

Forward compatibility

- Facts:
  - We are working hard to bring further improvements starting from our own apps
  - We know that some eSenders can and want to move faster than others
  - We chose (and promised) to maintain backward-compatibility with SDK-1
  - Further evolution of the regulatory domain is ongoing
- Questions:
  - How can we maintain our future-readiness?
  - What can we do to avoid holding back?
  - How can we make the next major SDK update (SDK-3) smooth and strong at the same time?

Forward compatibility

- Our answer to this challenge is **being “forward compatible”**
- Here is an example of how this will work:
  - Say we want to make important updates in the notice-type definition format to enable a wizard like interface and clean-up obsolete objects.
  - SDK-2 will contain a folder named **/forward**
  - In that folder we will publish the new format (under /forward/notice-types).
  - The new format will be equally as stable and equally as supported as the standard format.
  - eSenders that wish to move ahead and take advantage of such new features will be able to opt-in and use the new format without waiting for SDK-3.
  - When SDK-3 is eventually published, backward-compatibility with SDK-1 will be discontinued and everyone will move to an SDK-3 that will be tried and tested long before it is released.

SDK-2 - closing questions

THANK YOU

[OP-TED/eForms-SDK at efx-2]()

Publications Office of the European Union