# Designing a Static Analysis Tool for Elm



humio
a CrowdStrike company

Elm Radio

Jeroen Engels
@jfmengels

# elm-review

Creating guarantees

Communication

Reducing false positives

Ignoring reports

Trustworthy automatic fixes

# Static Code Analysis Tools

```
if (user.jobTitle = "manager") {
  // Do something...
}
```

# Linting



# Linting?

**linting**

NOUN

the process of making lint

*Collins English Dictionary*

Linters are about enforcing code style

# Code formatters

```elm
module MyModule exposing (Model, Msg, update, view)

import Html exposing (Html, button, div, text)
import Html.Events exposing (onClick)


type alias Model =
    { count : Int }


type Msg
    = Increment
    | Decrement


update : Msg -> Model -> Model
update msg model =
    case msg of
        Increment ->
            { model | count = model.count + 1 }

        Decrement ->
            { model | count = model.count - 1 }


view : Model -> Html Msg
view model =
    div []
        [ button [ onClick Increment ] [ text "+1" ]
        , div [] [ text <| String.fromInt model.count ]
        , button [ onClick Decrement ] [ text "-1" ]
        ]
```

# Creating guarantees in a project

We have a great product,
it's working well and it never crashes.

But we have an inconsistent UI.
Let's improve that.

```elm
-- Page/Checkout.elm
viewConfirmButton : Html msg
viewConfirmButton =
    Html.button
        [ Attr.style "height" "35px"
        , Events.onClick UserClickedOnConfirmButton
        ]
        [ Html.text "Confirm" ]
```

```elm
-- Page/Payment.elm
viewPayButton : Html msg
viewPayButton =
    Html.button
        [ Attr.style "height" "33px"
        , Events.onClick UserClickedOnPayButton
        ]
        [ Html.text "Pay" ]
```

```elm
module Ui.Button exposing (confirm, cancel, andSomeOtherButtons)

confirm : msg -> String -> Html msg

cancel : msg -> String -> Html msg
```

```elm
-- Page/Checkout.elm
import Ui.Button as Button
viewConfirmButton : Html msg
viewConfirmButton =
    Button.confirm UserClickedOnConfirmButton "Confirm"
```

```elm
-- Page/Payment.elm
import Ui.Button as Button
viewPayButton : Html msg
viewPayButton =
    Button.confirm UserClickedOnPayButton "Pay"
```

"We now have great modules for every UI element.

Now we are sure to have a consistent UI across the application."

# New pull request comes in...

```
view : Model -> Html Msg
view model =
    Html.button
        [ Attr.style "height" "34px"
        , Events.onClick UserClickedOnRemoveButton
        ]
        [ Html.text "Remove" ]
```

How do we prevent this from happening?

# Types...? Tests...?

```
view : Model -> Html Msg
view model =
    Html.button
        [ Attr.style "height" "34px"
        , Events.onClick UserClickedOnRemoveButton
        ]
        [ Html.text "Remove" ]
```

If you can detect the problem just by reading the code, static code analysis can help.

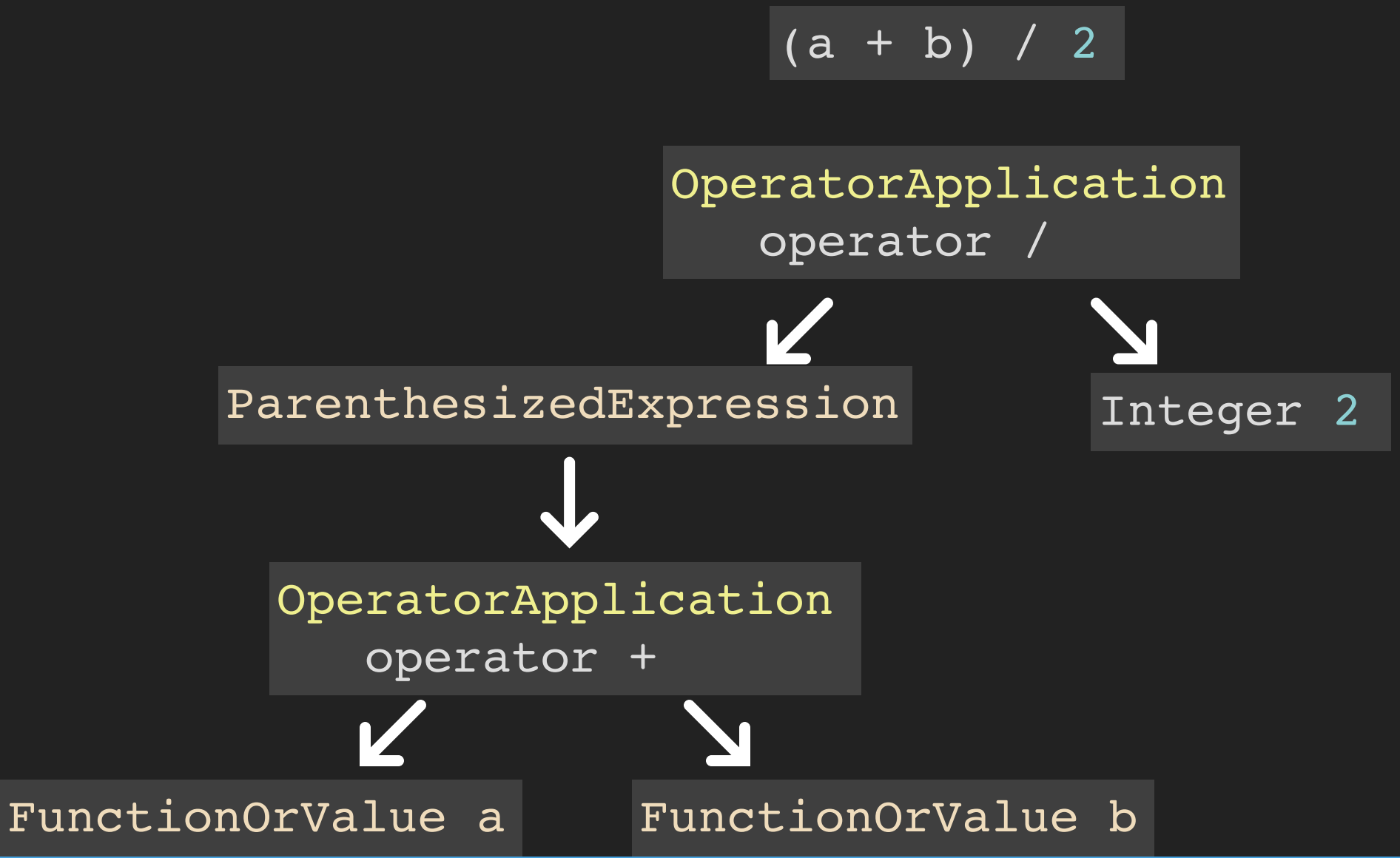
We need a linter
with custom rules

# Writing a Static Code Analysis Rule

# Written in Elm, for Elm

- Every Elm developer knows Elm
- No runtime errors

# Abstract Syntax Trees



```
type Declaration
    = FunctionDeclaration Function
    | AliasDeclaration TypeAlias
    | CustomTypeDeclaration Type
    | PortDeclaration Signature
    | InfixDeclaration Infix
```

```
type Expression
    = UnitExpr
    | Application (List (Node Expression))
    | OperatorApplication String InfixDirection (Node Expression) (Node Expression)
    | FunctionOrValue ModuleName String
    | IfBlock (Node Expression) (Node Expression) (Node Expression)
    | PrefixOperator String
    | Operator String
    | Integer Int
    | Hex Int
    | Floatable Float
    | Negation (Node Expression)
    | Literal String
    | CharLiteral Char
    | TupledExpression (List (Node Expression))
    | ParenthesizedExpression (Node Expression)
    | LetExpression LetBlock
    | CaseExpression CaseBlock
    | LambdaExpression Lambda
    | RecordExpr (List (Node RecordSetter))
    | ListExpr (List (Node Expression))
    | RecordAccess (Node Expression) (Node String)
    | RecordAccessFunction String
    | RecordUpdateExpression (Node String) (List (Node RecordSetter))
    | GLSLExpression String
```

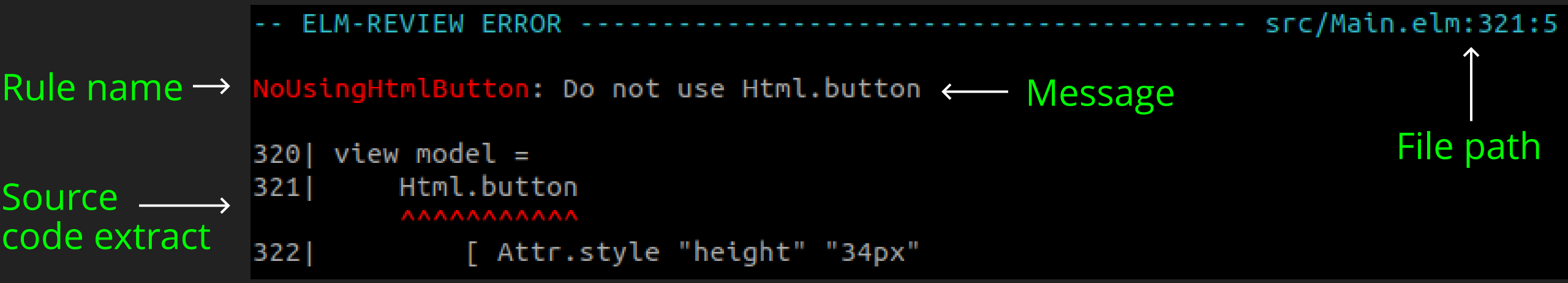
```elm
rule : Rule
rule =
    Rule.newSchema "NoUsingHtmlButton" ()
        |> Rule.withExpressionVisitor expressionVisitor
        |> Rule.fromSchema


expressionVisitor : Node Expression -> Context -> ( List Error, Context )
expressionVisitor node context =
    case Node.value node of
        Expression.FunctionOrValue [ "Html" ] "button" ->
            ( [ -- Report an error
              ]
            , context
            )

        _ ->
            ( [], context )
```

```
expressionVisitor : Node Expression -> Context -> ( List Error, Context )
expressionVisitor node context =
    case Node.value node of
        Expression.FunctionOrValue [ "Html" ] "button" ->
            ( [ Rule.error
                    "Do not use Html.button"
                    (Node.range node)
              ]
            , context
            )

        _ ->
            ( [], context )
```




# Are we getting the message across?

```
-- TYPE MISMATCH ---------------------------------------- src/Article/Feed.elm

The 2nd branch of this `if` does not match all the previous branches:

283|     if condition then
284|         1
285|
286|     else
287|         "2"
            ^^^
The 2nd branch is a string of type:

    String

But all the previous branches result in:

    number

Hint: All branches in an `if` must produce the same type of values. This way, no
matter which branch we take, the result is always a consistent shape. Read
<https://elm-lang.org/0.19.1/custom-types> to learn how to “mix” types.

Hint: Try using String.toInt to convert it to an integer?
```

# Developers need to understand

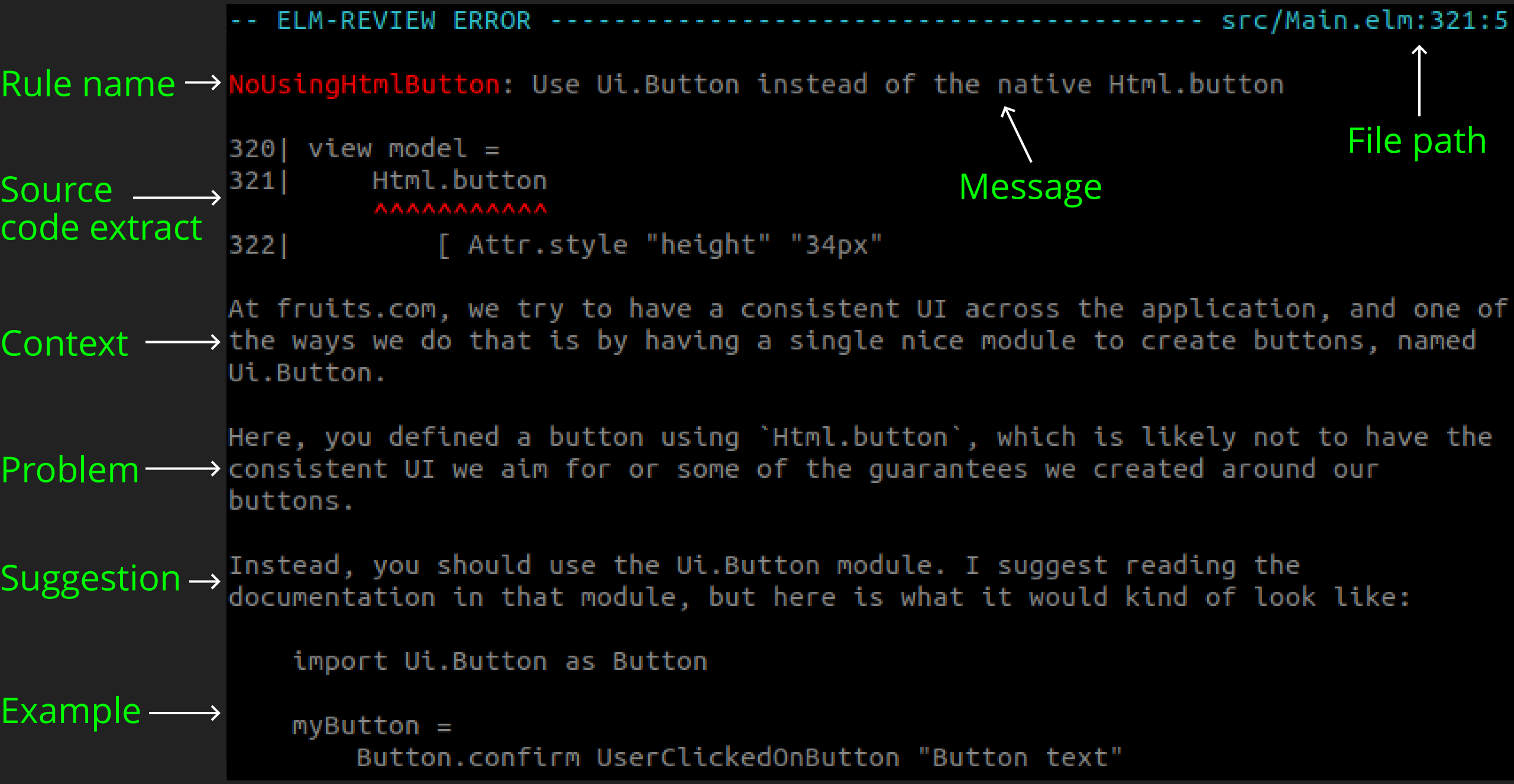
- What they did wrong
- Why it is a problem
- How to move forward

```
expressionVisitor : Node Expression -> Context -> ( List Error, Context )
expressionVisitor node context =
    case Node.value node of
        Expression.FunctionOrValue [ "Html" ] "button" ->
            ( [ Rule.error
                    "Use Ui.Button instead of the native Html.button"
                    (Node.range node)
              ]
            , context
            )

        _ ->
            ( [], context )
```

```elm
expressionVisitor : Node Expression -> Context -> ( List Error, Context )
expressionVisitor node context =
    case Node.value node of
        Expression.FunctionOrValue [ "Html" ] "button" ->
            ( [ Rule.error
                    { message = "Use Ui.Button instead of the native Html.button"
                    , details =
                      [ "At fruits.com, we try to have a consistent UI across the application
                      , "Here, you defined a button using `Html.button` or `Html.Styled.Butto
                      , "Instead, you should use the Ui.Button module. I suggest reading the
                      ]
                    }
                    (Node.range node)
              ]
            , context
            )

        _ ->
            ( [], context )
```


-- ELM-REVIEW ERROR ---------- src/Main.elm:321:5
NoUsingHtmlButton: Use Ui.Button instead of the native Html.button
320| view model =
321|     Html.button
        ^^^^^^^^^^^
322|         [ Attr.style "height" "34px"
At fruits.com, we try to have a consistent UI across the application, and one of the ways we do that is by having a single nice module to create buttons, named Ui.Button.
Here, you defined a button using `Html.button`, which is likely not to have the consistent UI we aim for or some of the guarantees we created around our buttons.
Instead, you should use the Ui.Button module. I suggest reading the documentation in that module, but here is what it would kind of look like:
import Ui.Button as Button
myButton =
    Button.confirm UserClickedOnButton "Button text"
Rule name
Source code extract
Context
Problem
Suggestion
Example
Message
File path

# Not understanding reports leads to workarounds

```elm
view : Model -> Html Msg
view model =
    -- elm-review-disable-next-line
    Html.button
        [ Attr.style "height" "34px"
        , Events.onClick UserClickedOnRemoveButton
        ]
        [ Html.text "Remove" ]
```

**linting**

NOUN

the process of making lint

*Collins English Dictionary*

Developers need to trust their tools

False positives

False negatives

Linters report lots of false positives

Missing information

↓

Assumptions

↓

False positives/negatives

# Providing information

- Type information
- Dependencies
- Project manifest, README
- Collecting data while traversing

# False positive in Ui.Button

```
module Ui.Button exposing (confirm, cancel, andSomeOtherButtons)
```

```
type alias Context =
  { isUiButtonModule : Bool }

rule : Rule
rule =
    Rule.newSchema "NoUnusedDeclarations" { isUiButtonModule = False }
        -- visitors...
        |> Rule.withModuleDefinitionVisitor moduleDefinitionVisitor
        |> Rule.fromSchema

moduleDefinitionVisitor : Node Module -> Context -> ( List Error, Context )
moduleDefinitionVisitor (Node _ { moduleName }) context =
    ( [], { isUiButtonModule = (moduleName == "Ui.Button") } )
```

```
expressionVisitor : Node Expression -> Context -> ( List Error, Context )
expressionVisitor node context =
    case Node.value node of
        Expression.FunctionOrValue [ "Html" ] "button" ->
            if not context.isUiButtonModule then
                ( [ Rule.error {- ... -} ]
                , context
                )

            else
                ( [], context )

        _ ->
            ( [], context )
```

# Gathering context from multiple files

# Report unused exports

```
module A exposing (used, neverUsed)

used =
    "some thing"

neverUsed =
    "other thing"
```

```
module B exposing (thing)

import A

thing =
    A.used ++ " is exposed"
```

# What can you deduce from the target language?

# Elm is knowable

No dynamic function calls

Functions always return the same type

No side-effects

No mutations

No macros

...

No memory management

No race conditions

...

Elm is knowable

↓

Better understanding of the code

↓

Almost no false positives

↓

No disable comments

# Ignoring reports

# Indeterminate rules

# Generated/vendored code

# Ignoring rules

```
config : List Rule
config =
    [ NoUnused.Variables.rule
    , NoDebug.Log.rule
    ]
        |> Rule.ignoreErrorsForDirectories [ "src-generated/", "src-vendor/" ]
```

# Allow existing errors

# Deprecating a function

```
{-| Does something.

@deprecated Use someBetterFunction which does it better.

-}
someFunction input =
    -- do something with input
```

# Deprecating a function

```
-------------------------------------------------------------- src/Api.elm:285:12

NoDeprecated: Found new usage of deprecated element

284|     List.map (\error -> field ++ " " ++ error) errors
285|         |> someFunction
               ^^^^^^^^^^^^

This element was marked as deprecated and should not be used anymore.

Please check its documentation to know the alternative solutions.

I found 7 errors in 1 file.
```

# Ignoring per file

```
config : List Rule
config =
    [ NoDeprecated.rule
        |> Rule.ignoreErrorsForFiles
            [ "src/Api.elm"
            , -- ...and other files
            ]
    ]
```

# Suppressing errors

*review/suppressed/NoDeprecated.json*

```
{
  "version": 1,
  "automatically created by": "elm-review suppress",
  "learn more": "elm-review suppress --help",
  "suppressions": [
    { "count": 7, "filePath": "src/Api.elm" }
  ]
}
```

# Deprecating code

```
I found no errors!

There are still 7 suppressed errors to address.
```

# Deprecating code

```
──────────────────────────────────────────────────────────── src/Api.elm:286:12

(unsuppressed) NoDeprecated: Found new usage of deprecated element

285|     List.map (\error -> field ++ " " ++ error) errors
286|         |> someFunction
               ^^^^^^^^^^^^

This element was marked as deprecated and should not be used anymore.

Please check its documentation to know the alternative solutions.

Errors marked with (unsuppressed) were previously suppressed, but you introduced
new errors for the same rule and file. There are now more of those than what I
previously allowed. Please fix them until you have at most as many errors as
before. Maybe fix a few more while you're there?

I found 8 errors in 1 file.
```

# Deprecating code

```
I found no errors!

There are still 6 suppressed errors to address, of which you fixed 1!
```

# Automatic fixes

# Automatic fixes




# code style

Automatic fixes



code formatters

# Fixes should be trustworthy

```
 index.js                    | 110 +++++-----
 lib/removeAccents.js        | 527 +++++++++++++++++++++++-----------------------
 test/fuzzyMatchingTest.js   | 414 ++++++++++++++++++------------------------
 test/readmeTest.js          | 114 +++++-----
 4 files changed, 586 insertions(+), 579 deletions(-)

diff --git a/index.js b/index.js
index 5daa5c4..5fa47c5 100644
--- a/index.js
+++ b/index.js
@@ -1,59 +1,67 @@
-var fuzzySet = require('fuzzyset.js');
-var removeAccents = require('./lib/removeAccents');
+const fuzzySet = require('fuzzyset.js');
+const removeAccents = require('./lib/removeAccents');

 function FuzzyMatching(items) {
-    var self = this;
-    self.set = fuzzySet();
-    self.itemMap = {};
-
-    if (items) {
-        items.forEach(function(item) {
-            self.add(item);
-        });
-    }
+       const self = this;
+       self.set = fuzzySet();
+       self.itemMap = {};
+
+       if (items) {
+               for (const item of items) {
+                       self.add(item);
+               }
+       }
 }
```

# Prompt for fixes

```
-- ELM-REVIEW ERROR ------------------------------------------ src/Api.elm:56:1

NoUnused.Variables: Top-level variable `someFunction` is not used

55| someFunction : String -> List String
56| someFunction value =
    ^^^^^^^^^^^^
57|     [ doSomethingWith value

You should either use this value somewhere, or remove it at the location I
pointed at.

I think I can fix this. Here is my proposal:

54|
55| someFunction : String -> List String
56| someFunction value =
57|     [ doSomethingWith value
58|     , doSomethingElseWith value
59|     ]
60|

? Do you wish to apply this fix? › (Y/n)
```

Use static analysis tools to create guarantees

# Communication

- What is the problem?
- Why is it a problem?
- How to solve the problem?

Remove false positives
by providing information

Simpler and knowable
languages allow better analysis

Better alternatives than

disable comments

to ensure rules are enforced

Tools can create trust

by reporting real problems and presenting them well

# Thank you!






Elm Radio

Jeroen Engels
@jfmengels