ToiletPaper #78

jambit WHERE INNOVATION WORKS

Elm

Von Andreas Scharf

✖ Problem

Du benötigst eine **statisch typisierte, rein funktionale Programmiersprache** für die Webentwicklung.

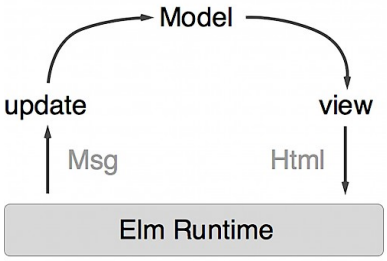
✔ Lösung

Evan Czaplicki (*@evancz*) begann 2012, **Elm** im Rahmen seiner Masterarbeit zu entwickeln. Elm ist eine funktionale Programmiersprache mit der Zielsetzung, Seiteneffekte zu vermeiden, einfach und performant zu sein. Der Code wird nach JavaScript übersetzt. Elm beinhaltet REPL, eine Paketverwaltung, einen Time-Traveling-Debugger, einen Entwicklungsserver und hochwertige Tools. Aufgrund statisch typisierter Prüfung verspricht das Projekt, dass in der Praxis keine Runtime Exceptions auftreten. Der Compiler ist bekannt dafür, nützliche Hinweise zu geben, Typen abzuleiten und Vorschläge für Tippfehler zu machen. Alle Werte in Elm sind immutable. Die Sprache hat ihr eigenes, recht kleines Modul-Ökosystem, welches eine semantische Versionierung basierend auf Library-Schnittstellen erzwingt. Beidseitige JavaScript-Interoperabilität ist möglich.

➔ Beispiel

Die **Elm-Architektur** ist ziemlich beliebt und ihr Design hat den State-Container Redux beeinflusst. Der Beispiel-Code rendert die Markdown-Inhalte des Benutzers in HTML und zeigt die typische Elm-Anwendungsstruktur mit Modell, View und Update.

- Das **Modell** definiert eine Datei, die den Anwendungsstatus beinhaltet.
- Die **Ansicht** ist eine Funktion, die das Modell als Input nimmt und HTML-Markup ausgibt und so den Anwendungsstatus und übersichtlich darstellt.
- Die **Update**-Funktion erhält das aktuelle Modell und eine Nachricht und gibt ein neues, aktualisiertes Modell aus.



Die sich daraus ergebende Anwendung ist eine Endlosschleife. Neue Eingaben seitens des Benutzers erzeugen eine Änderungsnachricht mit Inhalt. Die Update-Funktion ersetzt den Inhalt des Modells. Das neue Modell wird an die Ansicht-Funktion übergeben und mithilfe einer Markdown-Library in HTML gerendert.

```elm
main : Program () Model Msg
main = Browser.sandbox { init = initialModel , view = view , update = update }

-- MODEL
type alias Model = { content : String }

initialModel : Model
initialModel = { content = "# Headline" }

-- UPDATE
type Msg = Change String

update : Msg -> Model -> Model
update msg model =
    case msg of
    |   Change content -> { model | content = content }

-- VIEW

view : Model -> Html Msg
view model =
    div []
    |   [ textarea [ placeholder "Markdown", onInput Change, value model.content ] []
    |   , Markdown.toHtml [] model.content
    |   ]
```

Testen Sie das Beispielprogramm online unter folgendem Link: *https://ellie-app.com/43JYBXN4DQ9a1*

+ Links

- Projekt: http://www.elm-lang.org
- Anleitung: https://guide.elm-lang.org
- Online testen: https://ellie-app.com/new
- Kuratierte Liste: https://github.com/isRuslan/awesome-elm
- Elm-Architektur: https://dennisreimann.de/articles/elm-architecture-overview.html