Mini Course: Elm

Jeremie Gillet

Session 4: Web Applications

July 16, 2021

Javascript Interop

When pure Elm is not enough

- There are 3 ways too interact with Javascript: flags, ports and custom elements
- Today we will compile Elm to a Javascript file, and embed it in a HTML file

```
elm make src/SomeFile.elm --output=someFile.js
```

- This will produce a a JavaScript file that exposes an `Elm.SomeFile.init()` function
- You can add other JavaScript or CSS things in there too

Jumping right in: Flags

Start with an advantage

- Download everything from https://github.com/oist/mini-course-elm
- Open a terminal, cd into the folder “session4” and run

```
elm make src/Flags.elm --output=flags.js
```

- New thing:
  - We now use the first argument of `init`
  - The type of flags can be several things, prefer `Json.Encoder.Value`
- Exercise: update Clock.elm from session 3 to avoid the initial lag

Ports

Communicate with JS

```
elm make src/Websockets.elm --output=websockets.js
```

- We can now send and receive messages to JS using ports
- New things:
  - `module Websockets` => `port module Websockets`
  - Port output is a `Cmd`
  - Port input is a `Sub`
  - The type of messages can be several things, prefer `Json.Encoder.Value`
  - Trick: detect keystroke Enter with `Html.Events.on "keydown"`

Use Local Storage

Save values in the browser

```
elm make src/LocalStorage.elm --output=localStorage.js
```

- We can now save values in the browser that persist after refresh
- Exercise: Add two buttons "Save" and “Reset" that call ports only on user input.
- New things:
  - Using `Json.Encoder.Value` + decoders for port messages
  - `updateWithStorage` supersedes regular `update` to separate port and pure Elm code

Custom Elements

Embed JS in your Elm app

```
elm make src/CustomElements.elm –output=customElements.js
```

- Custom elements are a feature of the Web Components standard
- They encapsulate a functionality (written in JavaScript) into an HTML element
- Elm creates an HTML node with an ID and attributes that matches the custom element, and the custom element produces the HTML rendering
- New things:
  - `Input.radio`

Web applications

Control everything in a single page

- Up until now, we have been using `Browser.sandbox` and `Browser.element`, which create a `<div>` that you can embed in HTML

- We can go further with `Browser.document` and `Browser.application`, which create full websites

- With `Browser.application`, you can also control and parse URLs without having to reload the page

Browser.Document

Control the title too

- `Browser.element` => `Browser.document`
- This will create a full single page application, no longer embedded in a div of an HTML document
- Can be used with `elm reactor` or embedded in `document.html`
- New things:
  - View: `Html msg` => `Document msg`
  - No need for `<div id="elmApp"></div>` in HTML

Navigation

Managing URLS

- `Browser.document => Browser.application`
- Can be ran with `elm reactor` or compiled
- Example: session4 > src > Navigation.elm

- New things:
  - More arguments for `init`
  - `onUrlRequest`, `onUrlChange`
  - You can keep a single `Model` for multiple pages
  - `Browser.Navigation`
  - There are internal and external links

What else?

More resources

- Official guide: https://guide.elm-lang.org
- Slack, Discourse, Twitter, Reddit: https://elm-lang.org/community
- Elm creator: Evan Czaplicki https://github.com/evancz
- Check out Evan’s talks on YouTube. In particular: The life of a file