Introduction to Scala.js

It's still Scala after all

Anton Sviridov, 21 May 2024 @ Imperial College London

Agenda

- Who am I
- What is Scala.js
- Why choose Scala.js
- What are we building
- Introducing Laminar
- How does JavaScript tooling fit in all of this
- App V1 highlights
- App V2-V3 (🪄🦄 **AI** 🦄🪄): **live demo**
- Ecosystem
- Conclusion

Who am I

- Anton Sviridov, **keynmol** on Github, **velvetbaldmime** on Twitter
- Building software for money since 2006 (on and off)
- Longest tenure was at Disney – a lot of Disney+ is built on functional Scala
- Currently at Sourcegraph building intelligent code search in various languages
- Last exposure to JavaScript was around 2008, didn't go back since – until Scala.js arrived
- I have a blog with lots of posts about various Scala projects

https://blog.indoorvivants.com

Besom and Smithy4s on AWS - Scala 3 good
We write a Scala 3 web service that talks to AWS Comprehend servic
without using AWS SDK, through the power of Smithy4s. We then tak

Simple anti-toddler game with Scala Native a
To save my Slack and Discord from messages sent by a toddler, I nee
software. Let's build one with Raylib and Scala Native.

Scala Native and Swift: building Twotm8 Mac
Let's build a Swift UI (MacOS) client app for Twotm8, using both Swif

Smithy4s full stack (p.4): Frontend and closin
Where we build parts of our app's frontend and discuss the overall ex

Smithy4s full stack (p.3): Backend testing *202*
Where we identify meaningful levels of testing without any controvers
for integration testing

Smithy4s full stack (p.2): Backend and deplo
Where we set up HTTP server, Postgres access with Skunk and Flywa

What is Scala.js

- Optimising compiler from Scala to JavaScript – all modern Scala versions
- Gives you access to **all language features**, **most of Scala standard library, most of Java standard library**
  - With exceptions like multi-threading, date/time APIs, cryptography, etc.
- Has rich interop with JavaScript and various module systems (we'll be doing a **live demo** of that)
- Supports various JavaScript dialects
- Mature and battle-tested compiler

Why choose Scala.js?

- If you like Scala, you will **love** Scala.js – there are no restrictions on language features you can use! **It's still Scala after all**
- Mature, available in all Scala build tools
- Fast incremental linker
- Full access to JavaScript library ecosystem through interop features
- Great IDE support
- All the reasons why you'd choose Scala, with access to all the negatives of JavaScript!

Raw Scala.js

- Motivating example – creating a simple counter
- Using DOM APIs from Scala.js directly
- **Very** similar to reference JavaScript
- **Very** low level – easy to get lost in callbacks

```scala
import org.scalajs.dom.*

@main def hello =
  val container = document.getElementById("content")

  val button = document.createElement("button")
  button.innerText = "Click!"
  container.appendChild(button)

  val count =
    container.appendChild(document.createElement("h2"))
  container.appendChild(count)

  var counter = 250
  def setCounter() =
    counter += 1
    count.innerText = counter.toString()

  setCounter()

  button.addEventListener("click", ev => setCounter())
```

TODO app with Scala.js

This is not a simple TODO app - as you type out your new item, a list of similar TODO items will appear o

Generated with OpenAI

Investigate high costs on AWS to reduce expenses immediately.

Figure out what is costing me so much on AWS oh my god my family hasn't eaten in weeks

Similar todo notes:

Give my cousin my AWS credentials

💫🪄 Use AI magic to fill title 🪄💫

Create

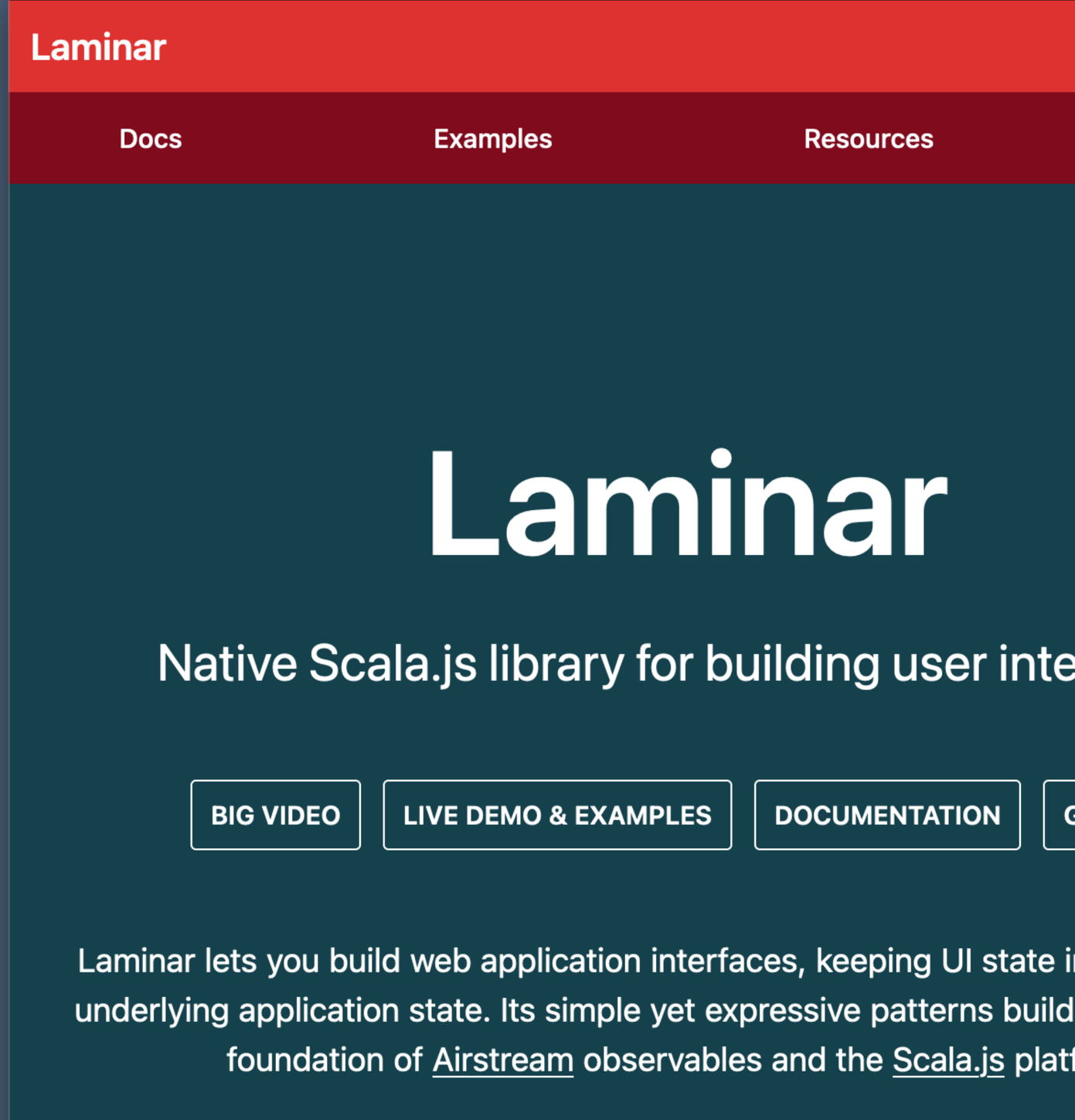
We're building a version of this app

Reopen Give my cousin my AWS credentials

Our project

A TODO app with Scala.js

- As TODO apps go, this one is pretty basic
- But even in V1 it has an extra feature not found among its competitors – search similar TODO items as you type
- We're doing it purely in browser, no backend, no database
- Main UI library we'll be using is Laminar
- **https://laminar.dev**
- Using JS tooling and JS libraries



Laminar

Motivating example - a simple counter

```scala
//> using platform scala-js
//> using dep com.raquo::laminar::17.0.0

import com.raquo.laminar.api.L.*

run
@main def hello =
  val counter = Var(0)
  val app = div(
    p(
      "count is ",
      child.text <-- counter,
      onClick --> { _ => counter.update(_ + 1) }
    )
  )
```

- Laminar is declarative
- Built on primitives like **Var**, **EventStream**, **Signal**
- Has lots of quiet syntax and helpers to keep programs concise

Laminar

Motivating example - a simple counter

```scala
//> using platform scala-js
//> using dep com.raquo::laminar::17.0.0

import com.raquo.laminar.api.L.*

run
@main def hello =
  val counter = Var(0)
  val app = div(
    p(
      "count is ",
      child.text <-- counter,
      onClick --> { _ => counter.update(_ + 1) }
    )
  )
```

Mutable reactive variable

Creating HTML elements (** **)

Adding text to element

Binding a text fragment to changes in reactive variable

Binding events (**onClick**) to a callback

85% of your frontend code will have nothing to do with web technologies

This is my (likely incorrect) thesis. Vast majority of the Cody you write will be about data manipulation and business logic.

So why not use a great language for that?

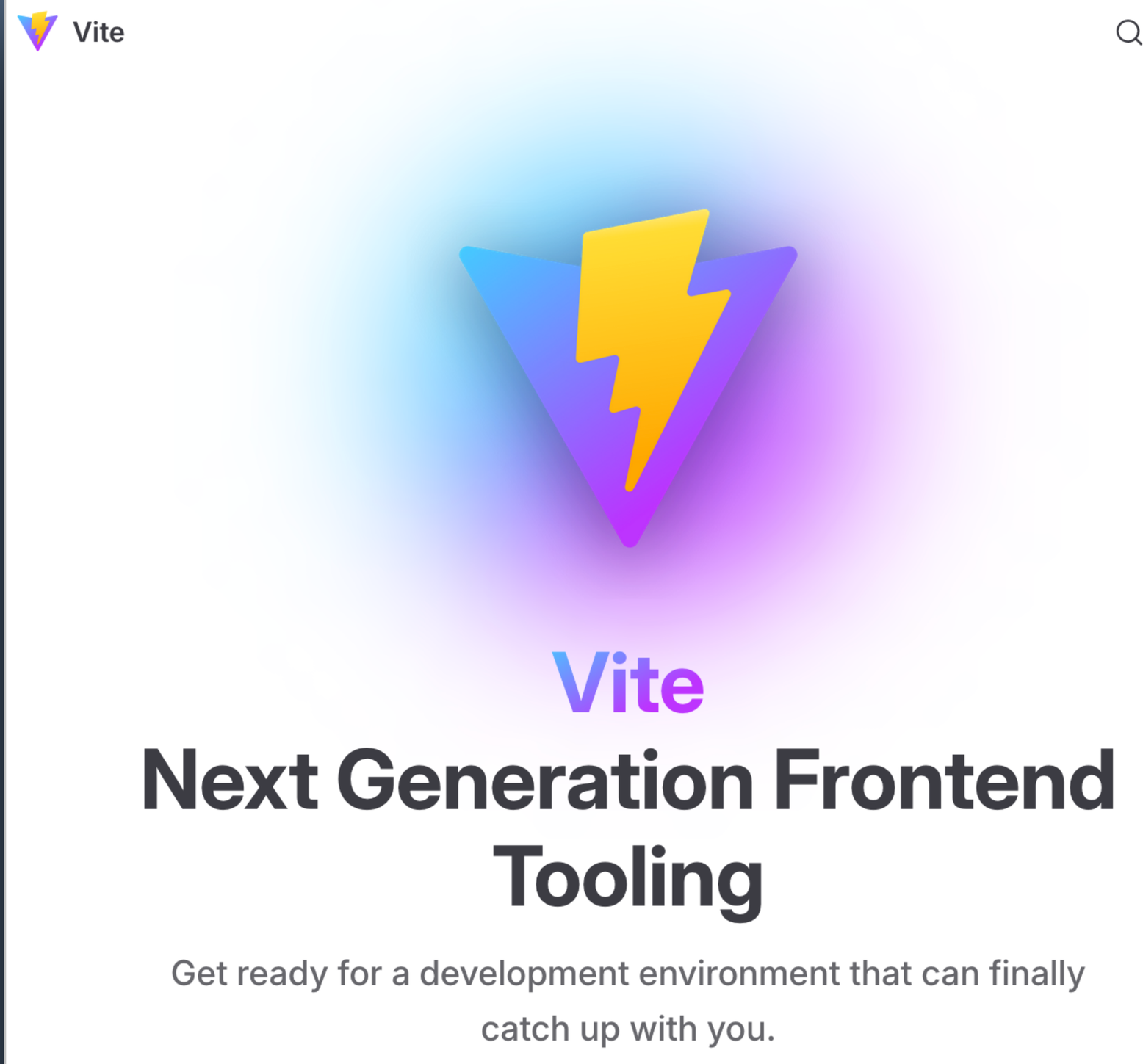
Frontend tooling from JavaScript world

TailwindCSS

- Modern CSS framework with utility classes
- Adds a whiff of professionalism to otherwise abysmal design
- **https://tailwindcss.com/**

Vite

- Modern backbone of JavaScript tooling
- Bundling, live reload, plugins, proxying etc.
- Voted “less likely to be abandoned within a month” in the JS build tools class
- **https://vitejs.dev/**



Getting started

A template for the quickest of quickstarts

https://github.com/keynmol/scalajs-scala-cli-vite-template

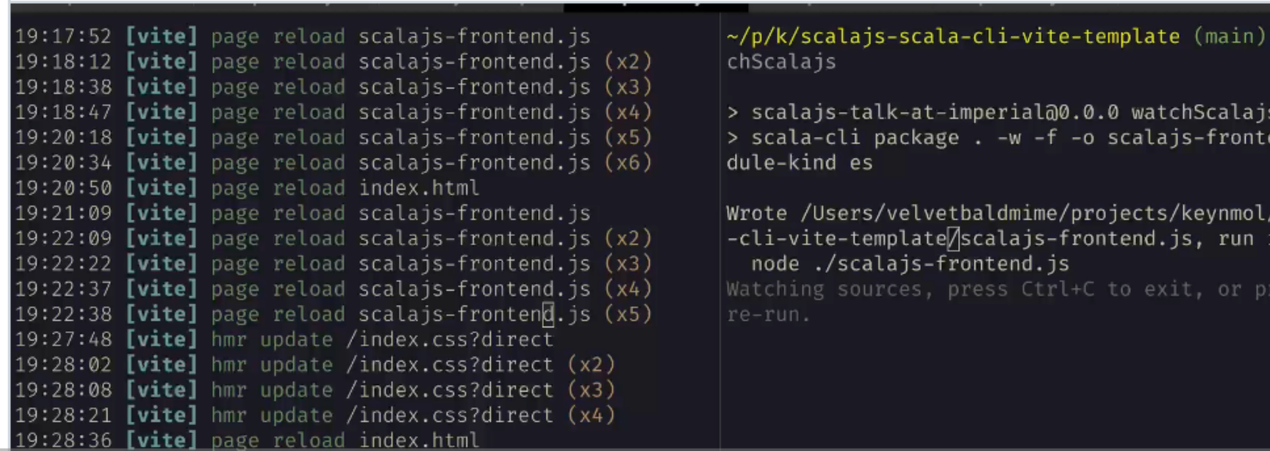
Scala.js ❤️ Scala CLI ❤️ Vite

Minimal template to get you started with Scala.js, Scala CLI and Vi

1. Run `npm install`
2. In two separate terminals:
   - `npm run watchScalajs` - for continuously rebuilding Scal
   - `npm run dev` - for running Vite's live server
3. Happy hacking! Your project will reload automatically as you m

Demo

CleanShot.2024-05-20.at.19.35.03.mp4



Building our app

Implementation plan

- Represent data and application state
- Continuously persisting state
- Components breakdown
- Sample components code
- Wiring it all together

Application data

- **derives ReadWriter** is a Scala 3 feature to derive a JSON codec for a case class
- Note TodoItem is only ever soft deleted
- SearchIndex references IDs from TodoIndex

```scala
import upickle.default.{ReadWriter}

case class TodoItem(
    title: String,
    description: String,
    deleted: Boolean = false
) derives ReadWriter

case class SearchIndex(
    inverted: Map[String, Map[Int, Float]],
    totalCount: Int
)

case class TodoIndex private (
    private val todos: Map[Int, TodoItem],
    private val lastId: Int
) derives ReadWriter
```

Application state

- SearchIndex will be automatically computed from TodoIndex
- We need both available to different components of the app (e.g. for listing and for searching)

```
1    case class AppState(
2        index: TodoIndex,
3        search: SearchIndex
4    )
```

Persisting state

```
7   object SaveState:
8     private val INDEX_KEY = "hello-scalajs"
9     private val DRAFT_KEY = "hello-scalajs-draft"
10
11    private def readFrom[T: ReadWriter](key: String, default: T) =
12      dom.window.localStorage.getItem(key) match
13        case null => default
14        case other =>
15          Try(read[T](other)).getOrElse(default)
16
17    private def save[T: ReadWriter](key: String, signal: Signal[T]) =
18      signal.map(upickle.default.write(_)) --> { written =>
19        dom.window.localStorage.setItem(key, written)
20      }
21
22    def restoreIndex(): TodoIndex =
23      readFrom[TodoIndex]("hello-scalajs", TodoIndex.empty)
```

Persisting state

```
7   object SaveState:
8     private val INDEX_KEY = "hello-scalajs"
9     private val DRAFT_KEY = "hello-scalajs-draft"
10
11    private def readFrom[T: ReadWriter](key: String, default: T) =
12      dom.window.localStorage.getItem(key) match
13        case null => default
14        case other =>
15          Try(read[T](other)).getOrElse(default)
16
17    private def save[T: ReadWriter](key: String, signal: Signal[T]) =
18      signal.map(upickle.default.write(_)) --> { written =>
19        dom.window.localStorage.setItem(key, written)
20      }
21
22    def restoreIndex(): TodoIndex =
23      readFrom[TodoIndex]("hello-scalajs", TodoIndex.empty)
```

Generic methods for anything serialisable into JSON

This creates a binder that we can attach to elements – process will continue until element is destroyed

Browser API for storing data in local storage

Persisting state

```
4    @main def hello =
5      val indexInit = SaveState.restoreIndex()
6      val state = Var(AppState(indexInit, SearchIndex.create(indexInit)))
7
8      val app =
9        div(
10         SaveState.saveIndex(state.signal.map(_.index)),
11         p("TODO app with Scala.js", cls := "text-6xl"),
```

Var#signal allows reacting to changes

Tailwind CSS classes look like this

- And so our app begins!
- Restore application state
- Create mutable Var
- Bind synchronisation process to the top-level ** ** element of the app

TODO app with Scala.js

This is not a simple TODO app - as you type out your new item, a list of similar TODO items will appear on the right

Investigate high costs on AWS to reduce expenses immediately.

Figure out what is costing me so much on AWS oh my god my family hasn't eaten in weeks

TodoItemForm.scala

💫🪄 Use AI magic to fill title 🪄💫

Create

Similar todo notes:

Give my cousin my AWS credentials

SimilarTodoItemsWidget.scala

TodoItemCard.scala

Reopen Give my cousin my AWS credentials

TodoItemList.scala

TodoItemCard

```
28    def TodoItemCard(id: Int, item: TodoItem, callback: (Int, Action) => Unit) =
29      def actionAndStyle(color: String, action: Action) =
30        List(
31          cls := s"m-2 text-md $color p-2 border-1 border-slate-700",
32          onClick --> { _ => callback(id, action) }
33        )
34
35      div(
36        idAttr := s"todo-$id",
37        cls := "bg-white rounded-md border-slate-300 p-4 border-1",
38        cls := (if item.deleted then "opacity-50" else "opacity-100"),
39        p(
40          if item.deleted then
41            button(
42              "Reopen",
43              actionAndStyle("bg-amber-200", Action.Restore)
44            )
45          else
46            button(
47              "Done",
48              actionAndStyle("bg-lime-200", Action.Delete)
49            )
50          ,
51          span(item.title, cls := "text-4xl"),
```

TodoItemCard

Invoke callback when button is clicked

Deleted items are 50% transparent

Render different button depending on state

Render item title

```
28    def TodoItemCard(id: Int, item: TodoItem, callback: (Int, Action) => Unit) =
29      def actionAndStyle(color: String, action: Action) =
30        List(
31          cls := s"m-2 text-md $color p-2 border-1 border-slate-700",
32          onClick --> { _ => callback(id, action) }
33        )
34
35      div(
36        idAttr := s"todo-$id",
37        cls := "bg-white rounded-md border-slate-300 p-4 border-1",
38        cls := (if item.deleted then "opacity-50" else "opacity-100"),
39        p(
40          if item.deleted then
41            button(
42              "Reopen",
43              actionAndStyle("bg-amber-200", Action.Restore)
44            )
45          else
46            button(
47              "Done",
48              actionAndStyle("bg-lime-200", Action.Delete)
49            )
50          ,
51          span(item.title, cls := "text-4xl"),
```

TodoItemList

- We pass Var if we plan to modify the state and Signal if we're only reading. There are safer way of doing this, but this will do
- We render a list of elements using **children <—** binder

```
3   def TodoItemList(
4     state: Var[AppState]
5   ) =
6     val actionUpdate =
7       (id: Int, act: Action) =>
8         state.update: state =>
9           act match
10            case Action.Delete =>
11              state.copy(index = state.index.delete(id))
12            case Action.Restore =>
13              state.copy(index = state.index.restore(id))
14
15    div(
16      cls := "grid gap-4 grid-cols-1",
17      children <-- state.signal
18        .map(_.index)
19        .map: todo =>
20          todo.items
21            .sortBy(_._1)
22            .reverse
23            .map(
24              TodoItemCard(_, _, actionUpdate)
25            )
26    )
```

```scala
@main def hello =
  val indexInit = SaveState.restoreIndex()
  val state = Var(AppState(indexInit, SearchIndex.create(indexInit)))

  val app =
    div(
      SaveState.saveIndex(state.signal.map(_.index)),
      p("TODO app with Scala.js", cls := "text-6xl"),
      p(
        "This is not a simple TODO app - as you type out your new item, a list of similar "
          + "TODO items will appear on the right",
        cls := "text-sm mx-2"
      ),
      TodoItemForm(state, "", ""),
      TodoItemList(state)
    )

  renderOnDomContentLoaded(
    dom.document.getElementById("hello"),
    app
  )
```

Once we have all of our individual components, the app comes together like this!

Live Demo time!

Scala.js Ecosystem

- **Tons** of Scala libraries are published for Scala.js and Scala JVM: **https://index.scala-lang.org/search?platforms=sjs1**
- Laminar is not the only game in town:
  - Use **Slinky** or **scalajs-react** if you prefer React
  - Use Outwatch or Calico if you prefer pure functional programming
- Play online with **https://scribble.ninja/**

Conclusion

- Scala.js is mature and cool
- Laminar is cool
- If you know Scala, you can be productive **immediately**
- If you don’t know Scala, it’s an exceptional vehicle to learn it
- If you know JS, give Scala.js a go, you already know most of the frontend tooling and gotchas!

Live version

https://scalajs-talk-at-imperial.fly.dev/

Code

https://github.com/keynmol/scalajs-talk-at-imperial