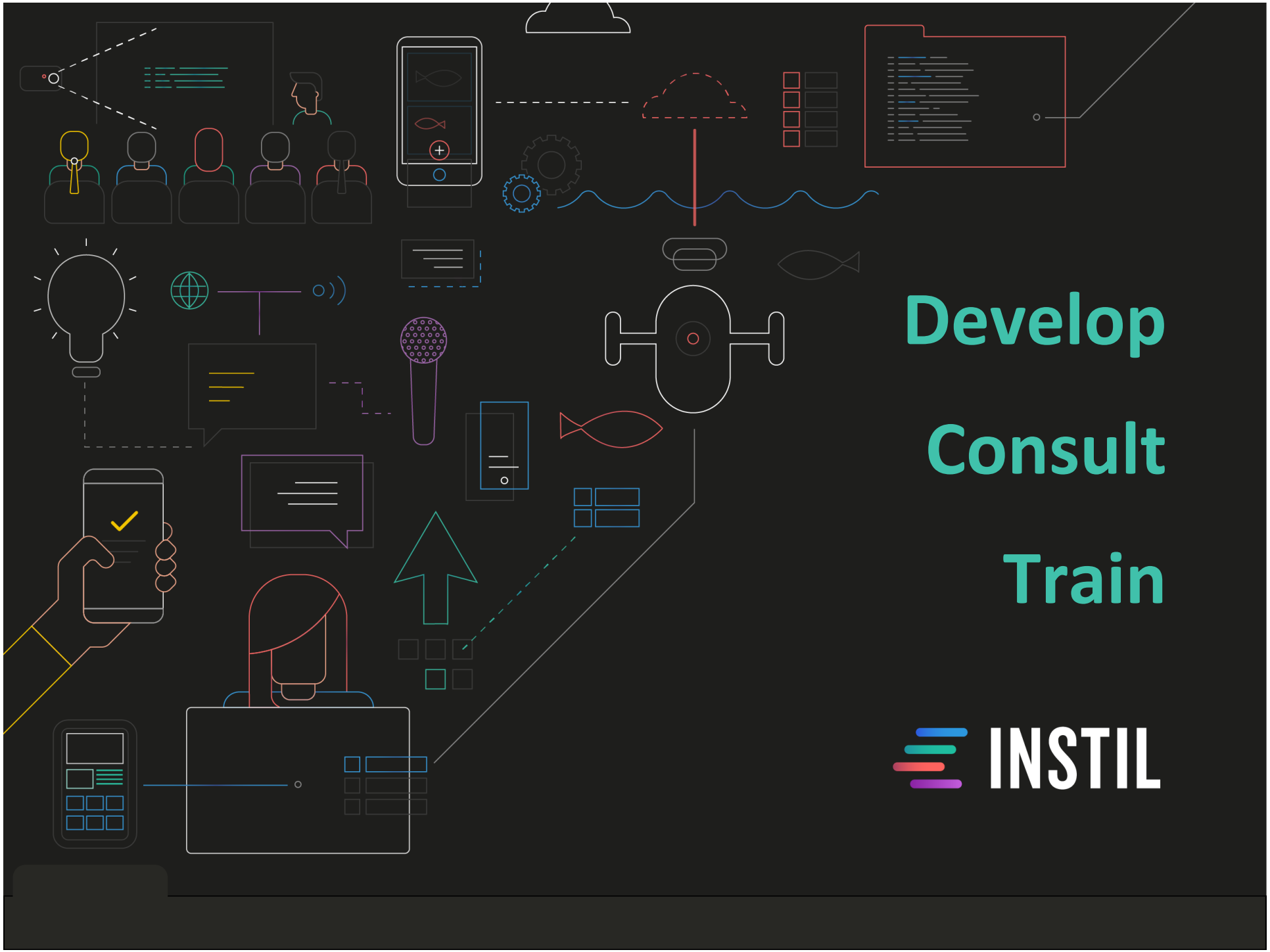
INSTIL

Down With JavaScript!

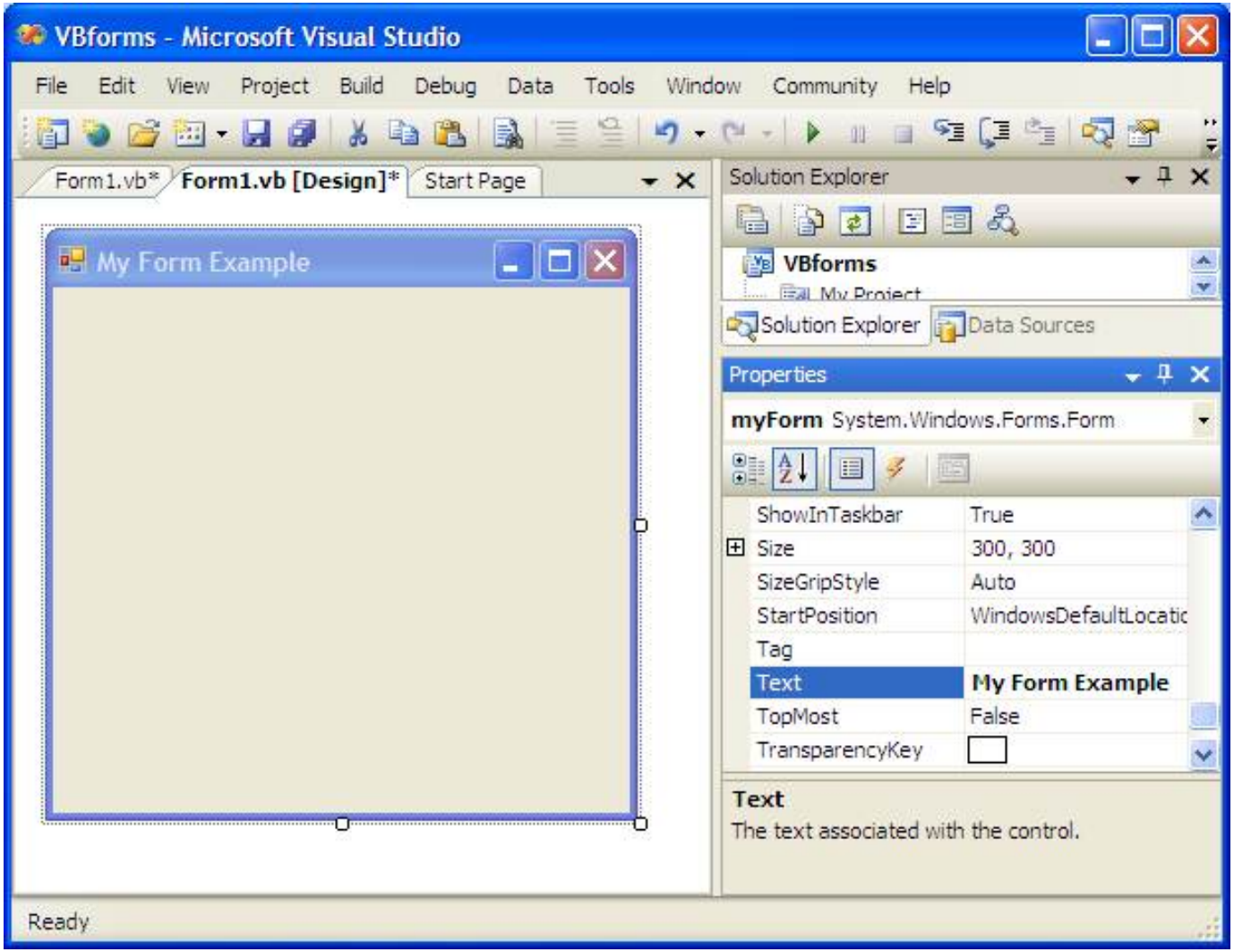
September 2018

training@instil.co

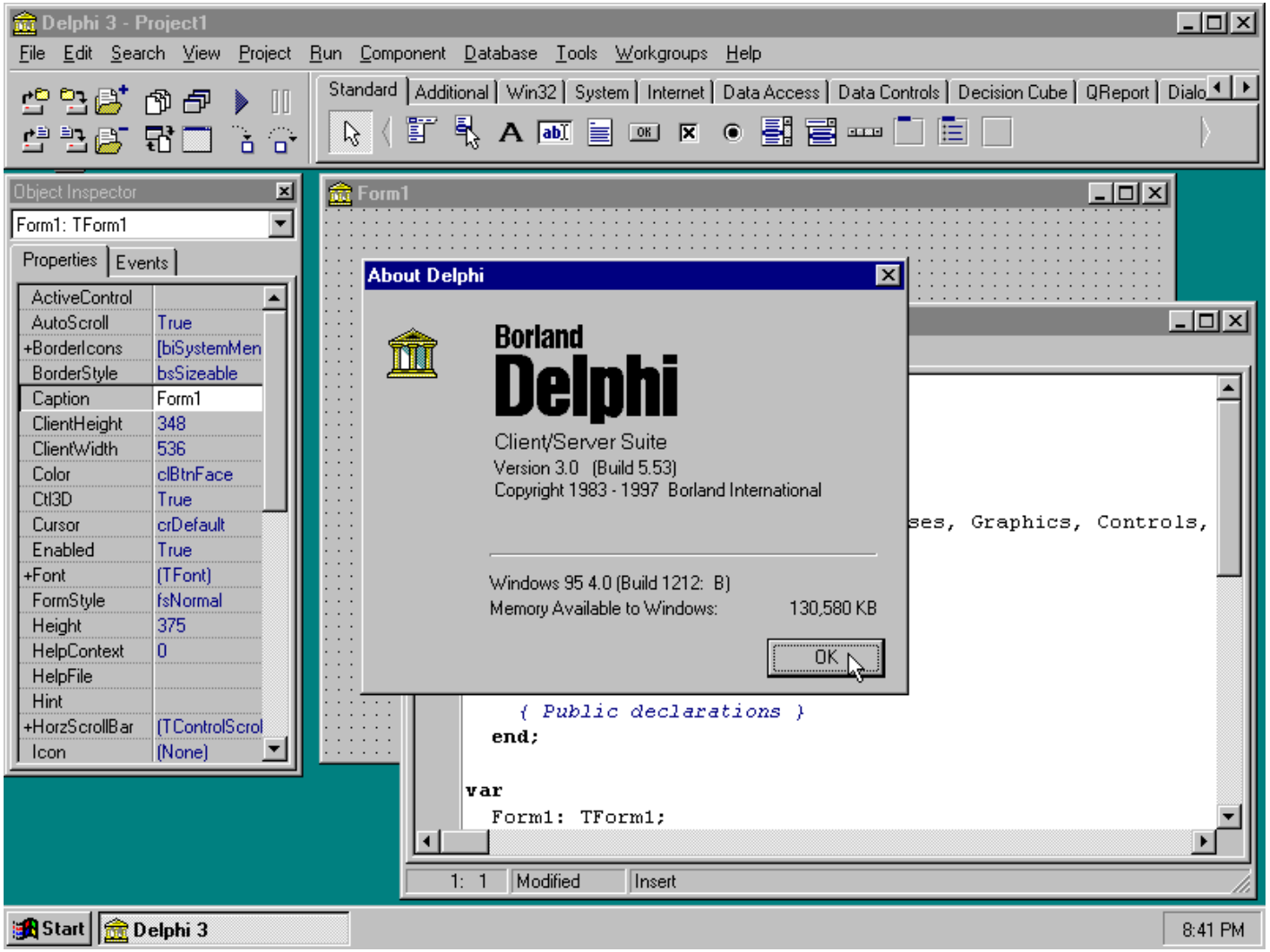

Develop
Consult
Train
INSTIL

Where Things Were Better…

Happiness is Visual Basic 6



True Happiness is Delphi



Imagine a Developer Was Frozen in the 1990's

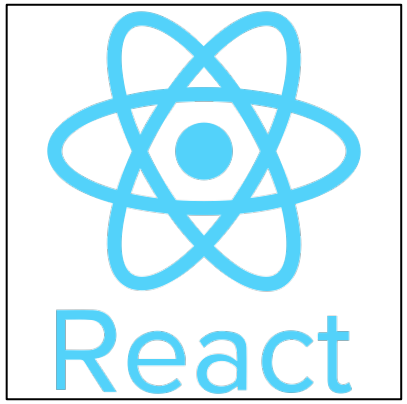
What Would They Expect?

What Do We Have Instead?

The Myth Of The Full Stack Developer…

But Salvation Is At Hand!!!

But Salvation Is At Hand!!!



React 101

<Ticker name="Fred"/>

```
class Ticker extends React.Component {
    constructor(props) {
        super(props);
        this.state = {secondsElapsed: 0};
    }
    componentDidMount() {
        this.interval = setInterval(() => this.tick(), 1000);
    }
    componentWillUnmount() {
        clearInterval(this.interval);
    }
    tick() {
        this.setState({secondsElapsed: this.state.secondsElapsed + 1});
    }
    render() {
        return (
            <div>
                <h1>Hello, {this.props.name}!</h1>
                <p>Page open {this.state.secondsElapsed} seconds.</p>
            </div>
        );
    }
}
```

```
class Ticker extends React.Component {
    constructor(props) {
        super(props);
        this.state = {secondsElapsed: 0};
    }
    componentDidMount() {
        this.interval = setInterval(() => this.tick(), 1000);
    }
    componentWillUnmount() {
        clearInterval(this.interval);
    }

    tick() {
        this.setState({secondsElapsed: this.state.secondsElapsed + 1});
    }
    render() {
        return (
            <div>
                <h1>Hello, {this.props.name}!</h1>
                <p>Page open {this.state.secondsElapsed} seconds.</p>
            </div>
        );
    }
}
```

Initial State

Lifecycle Methods

Changing state marks component as dirty

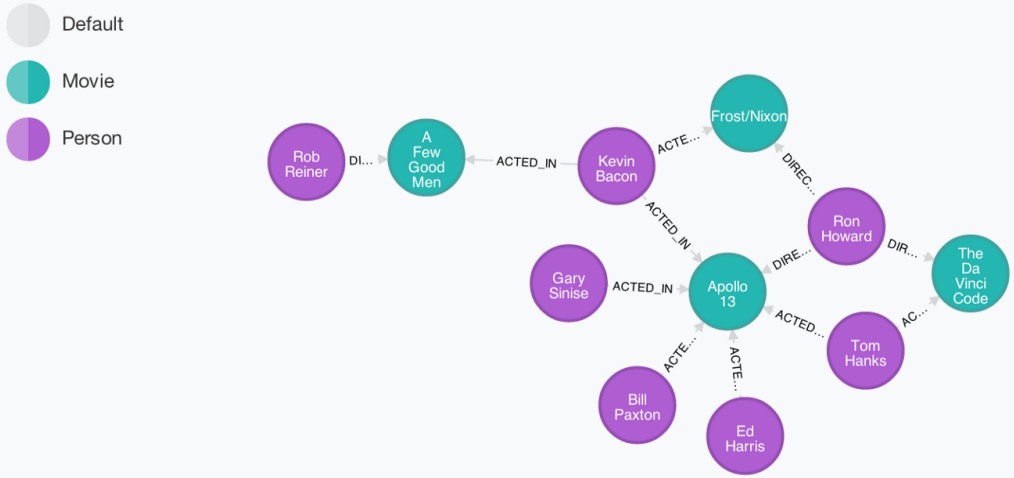
Rendering uses embedded JSX templating language

We Still Need Services and a Data Source

The Neo4J Desktop

```
1 match (kevin) -[:ACTED_IN]-> (movie) <-[:DIRECTED]- (director)
2 where kevin.name = 'Kevin Bacon'
3 return kevin, director, movie
```

CYPHER match (kevin) -[:ACTED_IN]-> (movie) <-[:DIRECTED]- (director) where kevin.name = 'Kevin Bacon' return



Displaying 11 nodes, 12 relationships

The Neo4J REST API

http://localhost:7474/

No Environment

POST | http://localhost:7474/db/data/transaction/commit | Params | Send | Save

Authorization | Headers (3) | Body | Pre-request Script | Tests | Cookies | Code

	KEY	VALUE	DESCRIPTION
☑	Accept	application/json	
☑	Content-Type	application/json	
☑	Authorization	Basic bmVvNGo6YWJjMTIz	
	Key	Value	Description

Bulk Edit | Presets

The Neo4J REST API

http://localhost:7474/

No Environment

POST http://localhost:7474/db/data/transaction/commit Params Send Save

Authorization Headers (3) Body Pre-request Script Tests Cookies Code

form-data x-www-form-urlencoded raw binary JSON (application/json)

```
1 {
2   "statements" : [ {
3     "statement" : "MATCH (m:Movie) RETURN m LIMIT 50"
4   } ]
5 }
```

Body Cookies Headers (5) Test Results Status: 200 OK Time: 173 ms Size: 39.28 KB

Pretty Raw Preview JSON

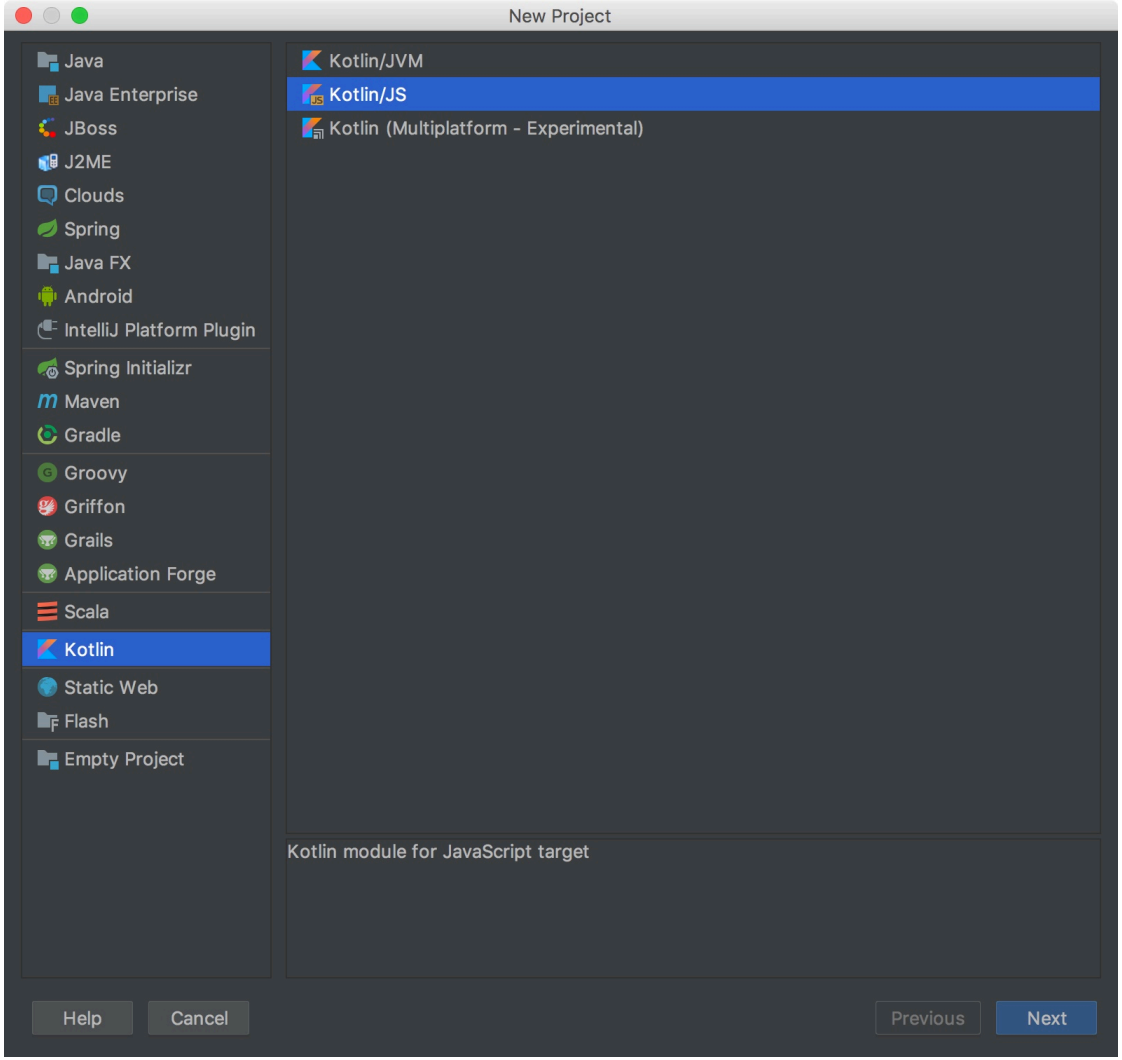

```
1 {
2     "results": [
3         {
4             "columns": [
5                 "m"
6             ],
7             "data": [
8                 {
9                     "row": [
10                        {
11                            "studio": "Twentieth Century Fox Film Corporation",
12                            "releaseDate": "1261090800000",
13                            "imdbId": "tt0499549",
14                            "runtime": 162,
15                            "description": "Disabled Marine Jake Sully travels to planet Pandora to become an avatar,
                                ingratiate himself with the natives and help Americans mine lucrative unobtainium. But
                                he finds himself in an interstellar conflict after falling for Na'vi warrior Neytiri.",
```

Demo No1

- Using Kotlin.js

INSTIL

Kotlin.js Support in IntelliJ



Kotlin.js React Support

GitHub, Inc.

TC build success | JB team

Create React Kotlin App

Create React apps in Kotlin with no build configuration.

Please note that this is an **early preview version**.

Quick Overview

Make sure you have JDK 8 installed before proceeding. Java 9 is not supported yet.

Create a new project:

```
npx create-react-kotlin-app my-app
```

(npx comes with npm 5.2+ and higher. See Installation for older npm versions.)

Run the project:

```
cd my-app/
npm start
```

Then open http://localhost:3000/ to see your app.
When you're ready to deploy to production, create a minified bundle with `npm run build`.

Create React Kotlin App will configure Kotlin, webpack, and IntelliJ IDEA so that you can focus on the code instead.

Just create a project, and you're good to go.

Kotlin Wrappers Repo

GitHub, Inc.

Kotlin Wrappers

This repository hosts a number of Kotlin wrappers for popular JavaScript libraries, such as React, Mocha, and styled-components.

To learn more please refer to the READMEs of individual modules:

- kotlin-css,
- kotlin-extensions,
- kotlin-mocha,
- kotlin-react,
- kotlin-react-dom,
- kotlin-react-redux,
- kotlin-react-router-dom,
- kotlin-redux,
- kotlin-styled.

All wrappers are available on npm: `npm install @jetbrains/<wrapper name>`

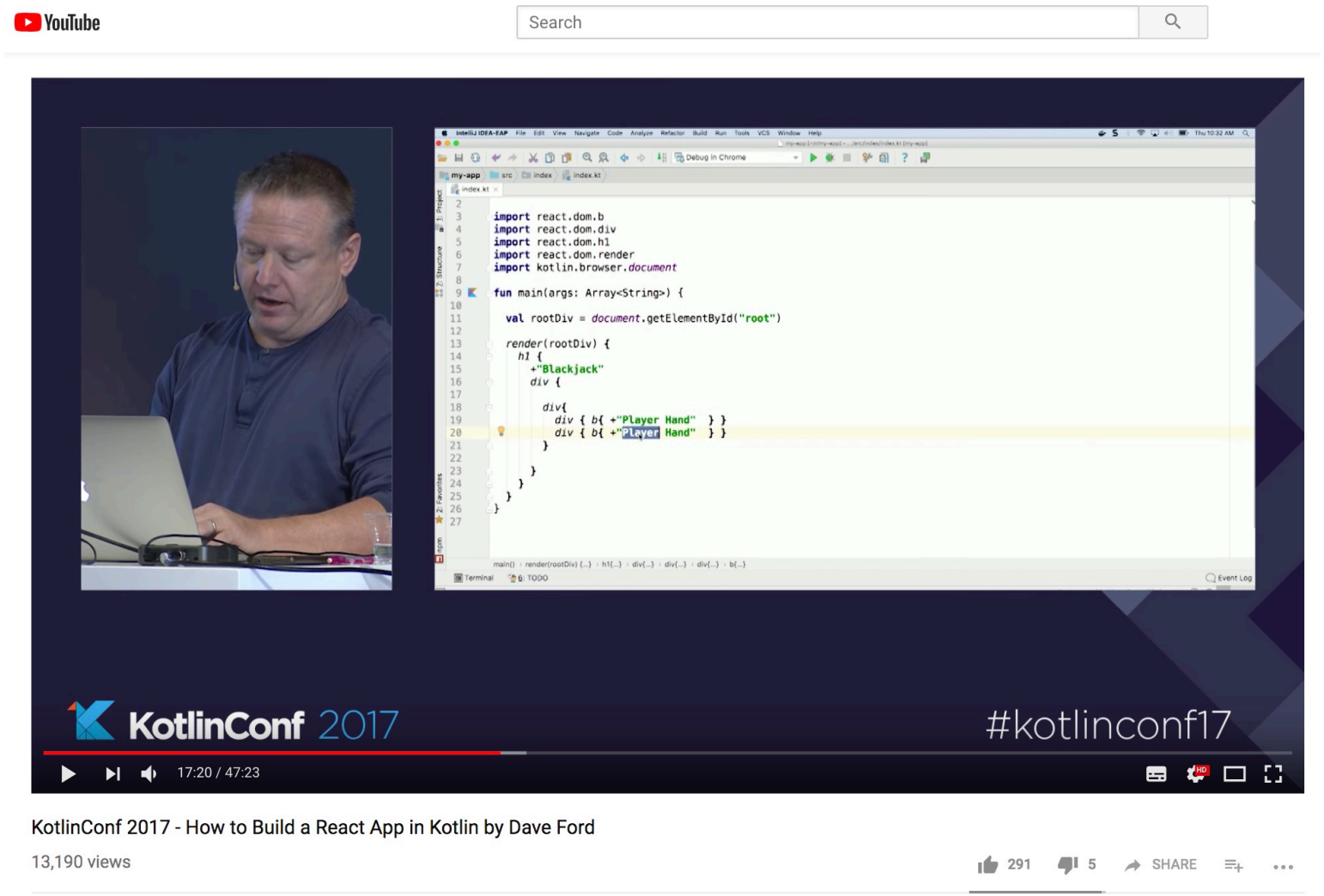
Artifacts installable using Maven and Gradle are published to Bintray.

All packages require JDK 8 to be installed.

Examples

1. To-do list example
2. Tic-Tac-Toe example
3. A port of "Thinking in React" example

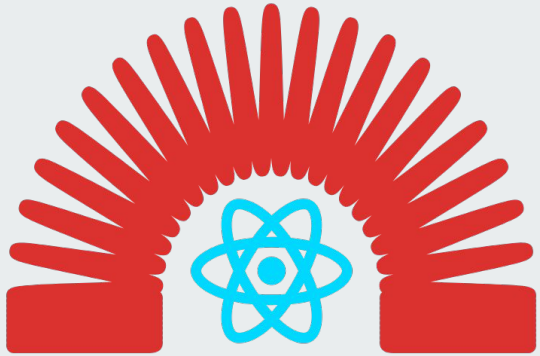
Kotlin Conf Presentation



The Demo…

Summing Up

Scala js with Slinky



Flames

Flames

What is Scala JS

- ➔ Scala code compiled to JS
- ➔ Full Scala support (apart from reflection)
- ➔ Fast dev feedback loop
- ➔ Compact production JS code generated
- ➔ Performance within 1-3x as optimised JS
- ➔ Type safety across multiple platforms

Scala for JS Devs

```
var xhr = new XMLHttpRequest()

xhr.open("GET",
 "https://api.twitter.com/1.1/search/" +
 "tweets.json?q=%23scalajs"
)
xhr.onload = (e: Event) => {
 if (xhr.status == 200) {
   var r = JSON.parse(xhr.responseText)
   $("#tweets").html(parseTweets(r))
 }
}
xhr.send()
```

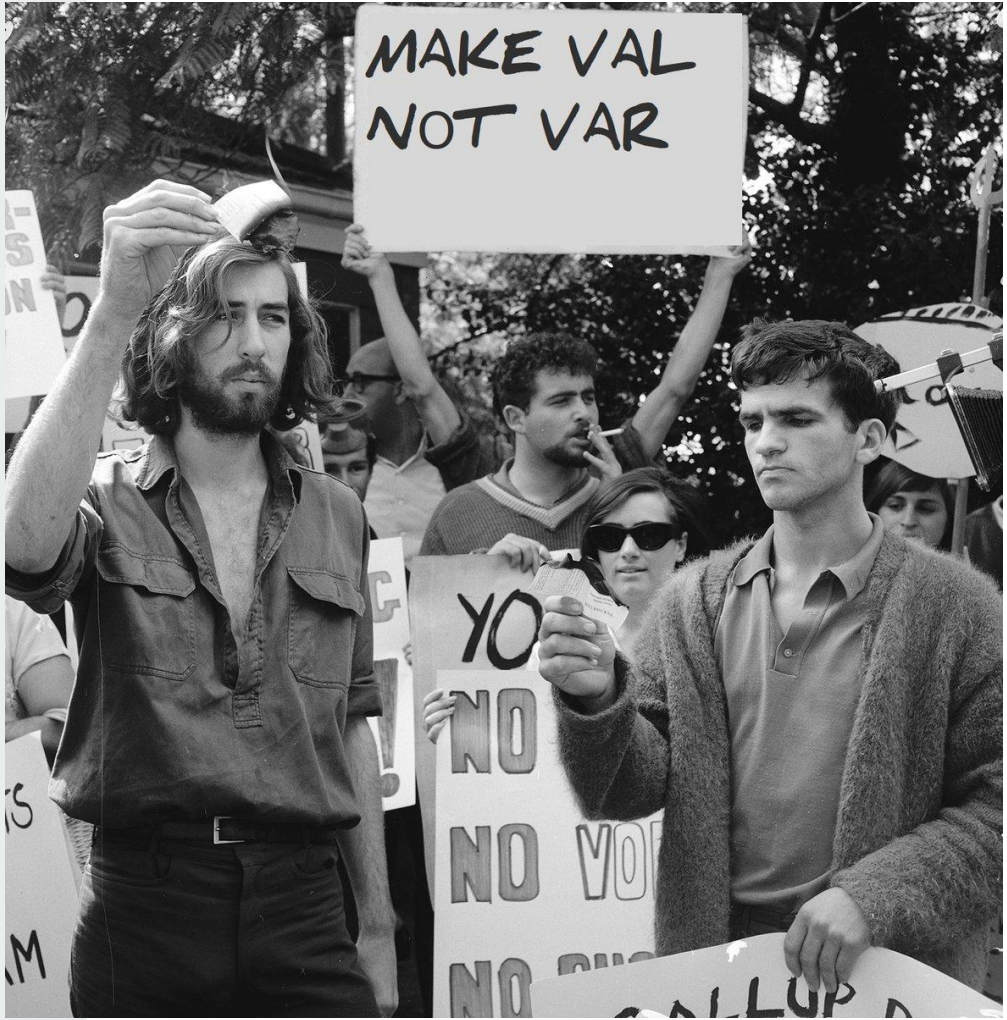
Ref: scala-js.org

JS Comparison: Variables

```
// mutable variable
let x = 5;
// immutable variable
const y = "Constant";
```

```
// mutable variable
var x = 5
// immutable variable
val y = "Constant"
```

Prefer Immutability



JS Comparison: Functions

```
function mult(x, y) {
  return x * y;
}
```

```
def mult(x: Double,
y: Double): Double = x * y
```

JS Comparison: Anonymous functions

```js
const f = (x, y) => x + y;

const p = ["Fox", "jumped",
"over", "me"];


const l = p.map(s => s.length)
  .reduce((a, b) => a + b, 0);
// == 15
```

```scala
val f = (x: Double,

    y: Double) => x + y

val p = Array("Fox", "jumped",
"over", "me")


val l = p.map(s => s.length)
  .foldLeft(0)((a,b) => a + b)
// == 15
```

JS Comparison: Higher order functions

```
function minmaxBy(arr, f) {
  return arr.reduce(
    ([min, max], e) => {
      const v = f(e);
      return [Math.min(min, v),
Math.max(max, v)]
    },
    [Number.MAX_VALUE,Number.MIN_VALUE]
  )
}
const [youngest, oldest] =
minmaxBy(persons, e => e.age);
```

```
def minmaxBy[T](seq: Seq[T],
    f: T => Int): (Int, Int) = {
  seq.foldLeft((Int.MaxValue,
Int.MinValue)) {
    case ((min, max), e) =>
      val v = f(e)
      (math.min(min, v),
        math.max(max, v))
  }
}
val (youngest, oldest) =
minmaxBy[Person](persons, _.age)
```

JS Comparison: Futures

```
function onLoadPromise(img) {
 if (img.complete) {
    return Promise.resolve(img.src);
 } else {
  const p = new Promise((success)=> {
     img.onload = (e) => {
      success(img.src);
     };
    });
return p;
   }
}
```

```
def onLoadFuture(img:
HTMLImageElement) = {
  if (img.complete) {
    Future.successful(img.src)
  } else {
    val p = Promise[String]()
    img.onload = { (e: Event) =>
      p.success(img.src)
    }
    p.future
  }
}
```

Other features

- Traits (Mixins)
- Singleton Objects
- Immutable Collections
- Case Classes
- Tuples
- Pattern Matching / destructuring
- Default parameter values
- Lazy Initialization
- Reified Generics
- TypeClass Support
- Higher Kinded Types Support
- Macros

…….

JavaScript Interop : Facades

```
@js.native
trait Window extends js.Object {
  val document: HTMLDocument = js.native
  var location: String = js.native

  def innerWidth: Int = js.native
  def innerHeight: Int = js.native
  …
}
```

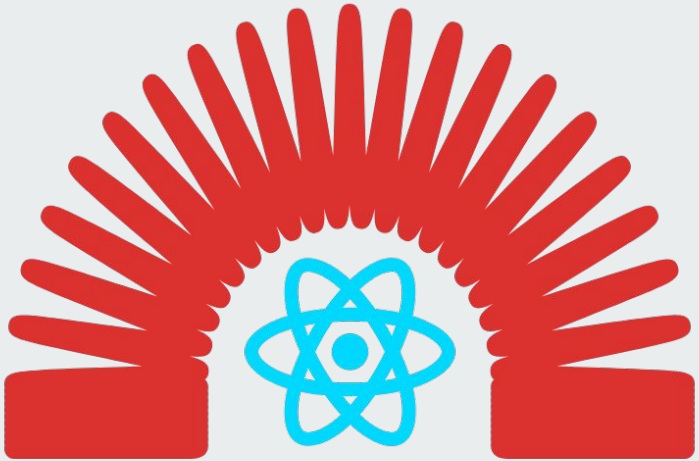
Typescript definitions to Scala.js Binding transformer available
https://github.com/sjrd/scala-js-ts-importer

JavaScript Interop : Exposing Scala.js

```
package example
import scala.scalajs.js.annotation._
@JSExportTopLevel("HelloWorld")
object HelloWorld {
  @JSExport
  def sayHello(): Unit = {
    println("Hello world!")
  }
}
HelloWorld.sayHello();
```

Also `@JSExportTopLevel("..")` to export a class

Slinky



- Scala JS wrapper for React
- Stays close as possible to React conventions
- Excellent integration with existing react components

React Stateless Component

```
@react class HelloMessage extends StatelessComponent {
  case class Props(name: String)

  override def render(): ReactElement = {
    div("Hello ", props.name)
  }
}

ReactDOM.render(
  HelloMessage(name = "Taylor"),
  mountNode
)
```

React Stateless Component expanded

```
object Hello extends StatelessComponentWrapper {
  case class Props(name: String)

  @ScalaJSDefined
  class Def(jsProps: js.Object) extends Definition(jsProps) {
    override def render() = {
      div("Hello ", props.name)
    }
  }

  def apply(name: String) = apply(Props(name))
}
```

Slinky: getting started

```
sbt new shadaj/create-react-scala-app.g8
```

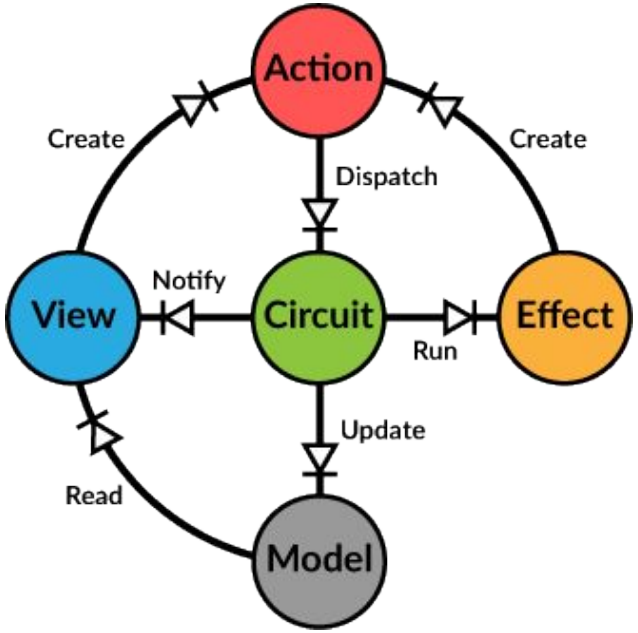
```
[IJ]sbt:movie-app> fastOptJS::startWebpackDevServer
```

```
~fastOptJS
```

- Dependent on Java 8, SBT, NPM

Demo

Diode



- Immutable application state
- Controlled data flow
- Similar to react
- React components integration

ScalaCSS: Typesafe CSS DSL

```scala
object MyStandalone extends StyleSheet.Standalone {
  import dsl._

  "div.std" - (
    margin(12 px, auto),
    textAlign.left,
    cursor.pointer,

    &.hover -
      cursor.zoomIn,

    &("span") -
      color.red,

    (media.tv.minDeviceAspectRatio(4 :/: 3) & media.all.resolution(300 dppx)) -
      width(600 px),

    media.not.handheld.landscape.color -
      width(500 px)
  )

  "h1".firstChild -
    fontWeight.bold
```

Slinky React Native

```
sbt new shadaj/create-react-native-scala-app.g8
```

```
npm run ios
```

```
~fastOptJS
```

- Dependent on SBT and Java 8, NPM, xcode

INSTIL

F# and Fable

Functional Client Side Development

September 2018

training@instil.co

Eamonn.Boyle@instil.co

@BoyleEamonn

Computer

F#

.NET strongly & statically typed multi-paradigm language

- Imperative
- Object Oriented
- Functional First

Open source, cross platform

- Mono, .NET Framework & .NET Core

Stable – currently on version 4.5 (August 2018)

- Version 1.0 released in 2005
- Passionate Community

A lot of nice C# features originated in F#

F# Syntax

Originally a .NET implementation of OCaml so the syntax is similar

Heavy use of type inference

- Expressions, function parameters and return types etc all inferred from usage
- Can reduce the amount of typing (pun intended)
- Can lead to very strange error messages

Focuses on pushing runtime errors to compile time

- Very strict type system

Easy interop with .NET (C#) libraries

```
open Types

unit -> Model * Cmd<Msg>
let init () : Model * Cmd<Msg> =
  "", []

Msg -> 'a -> Model * Cmd<Msg>
let update msg model : Model * Cmd<Msg> =
  match msg with
  | ChangeStr str -> str, []
```

Fable

Fable is an F# to JavaScript compiler powered by Babel, designed to produce readable and standard code.

Type bindings and helpers for easy interop with JavaScript

- Including Type Safe helpers for React

Supports most of the F# core library

There's a tool to convert TypeScript bindings to Fable bindings

- https://github.com/fable-compiler/ts2fable

Elmish

Elm-like abstractions for F#

Facilitates creating Model-View-Update applications

- Similar to using Redux with something like React
- Does not include the View part, instead you use the Fable React helpers

The main building blocks are

- Model
- Messages
- Commands
- Initialisation
- Update
- Views
- Program

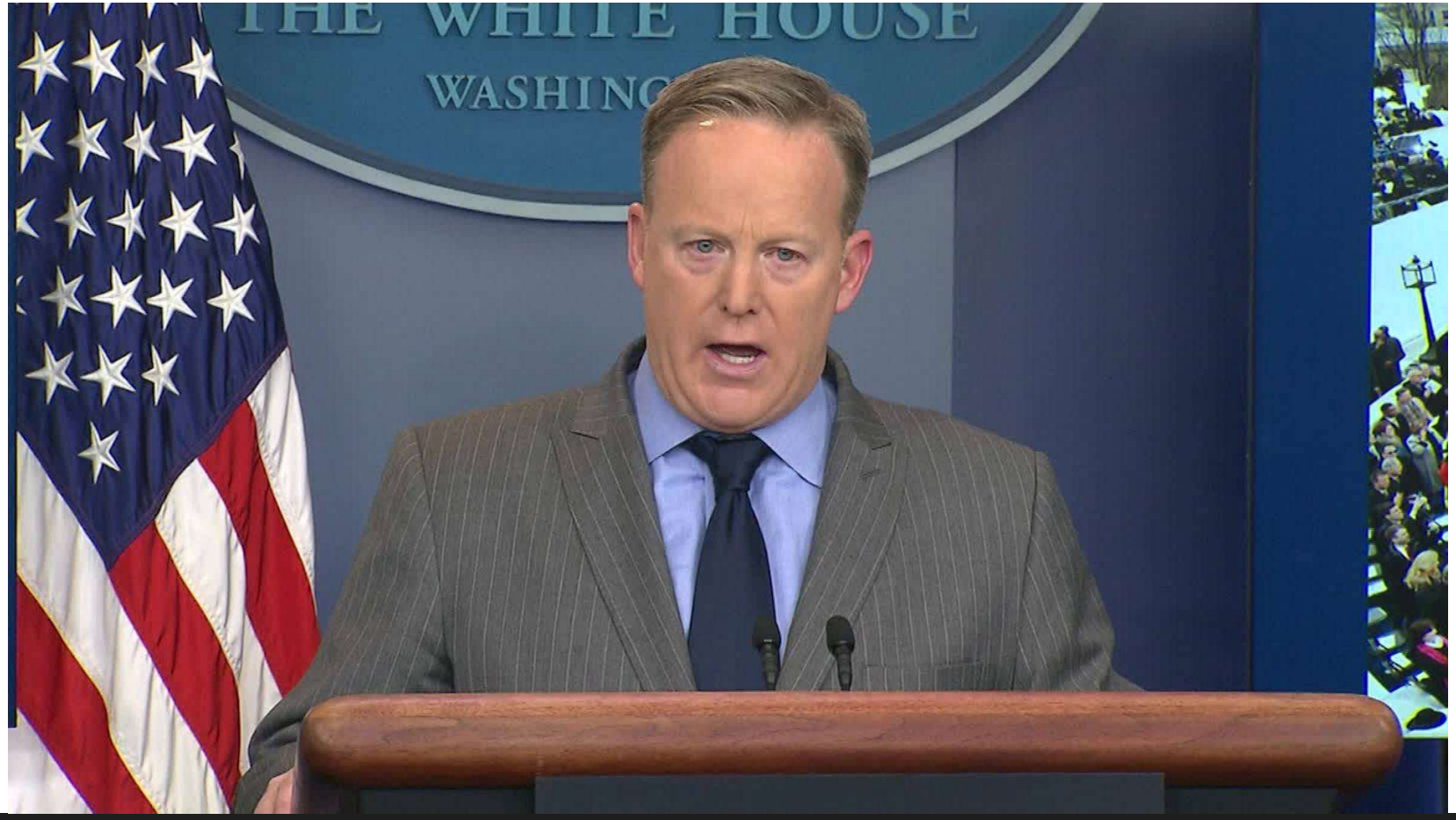
Demo

INSTIL

The Bad Stuff

INSTIL

"This was the hardest development experience, period!!"



The Bad Stuff (very subjective)

Type inference can be very confusing

- I enjoy type inference in C#, Kotlin etc
- This is at another level
- Inference in one part of the program can be dictated by usage far away
- This can lead to confusing error messages
- Explicit types can help here
- But then the syntax can be longer than other languages

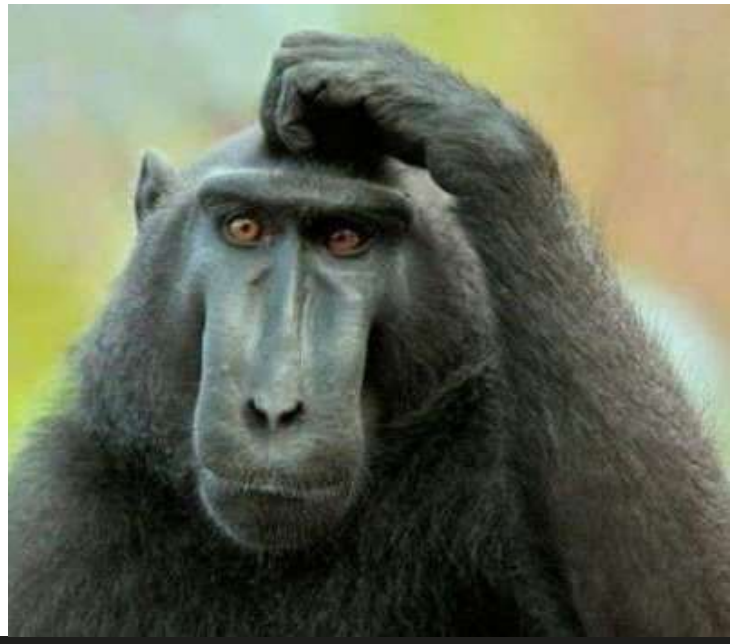
File order matters – this caught me out a few times

Tooling is not perfect

- I found myself moving between VS Code and JetBrains Rider

Runtime errors can still occur and can be difficult to debug

- Some of the error messages are minimal



The Bad Stuff (very subjective)

I found the React bindings a pain compared to using HTML

- It's better because it's type safe but…
- Tooling around HTML + CSS is pretty mature while this isn't
- More difficult if working with other tools, designers and existing skills
- I find standard JSX with React much easier

White space significance is a pain

I found it difficult to structure the app

- Some blocks of code were getting very large
- I would need to build much larger apps to see how this scales

The Good Stuff

INSTIL

The Good Stuff

The language is pretty cool - succinct functional syntax

- Pipes
- Function Composition
- Type Definition and Domain Modeling
- Operator Overloading
- etc

Strict Type System

- Catches a lot of errors
- Exhaustive pattern matching
- Units of measure
- Etc

Easy interop and drop down to OO or imperative style if required

The Good Stuff

This ability to use OO & imperative is essential when targeting JavaScript platform

- Good bindings to access dynamic types

Tooling is very good

- Very quick to create a project from templates
- Visual Studio Code + Ionide is pretty good and is completely free
- HTML to Elmish converter - https://mangelmaxime.github.io/html-to-elmish/
- Hot Module Reloading while retaining state

Elmish MVU pattern is very good and is well implemented

- Very succinct to build up types, state and views

Conclusion

INSTIL

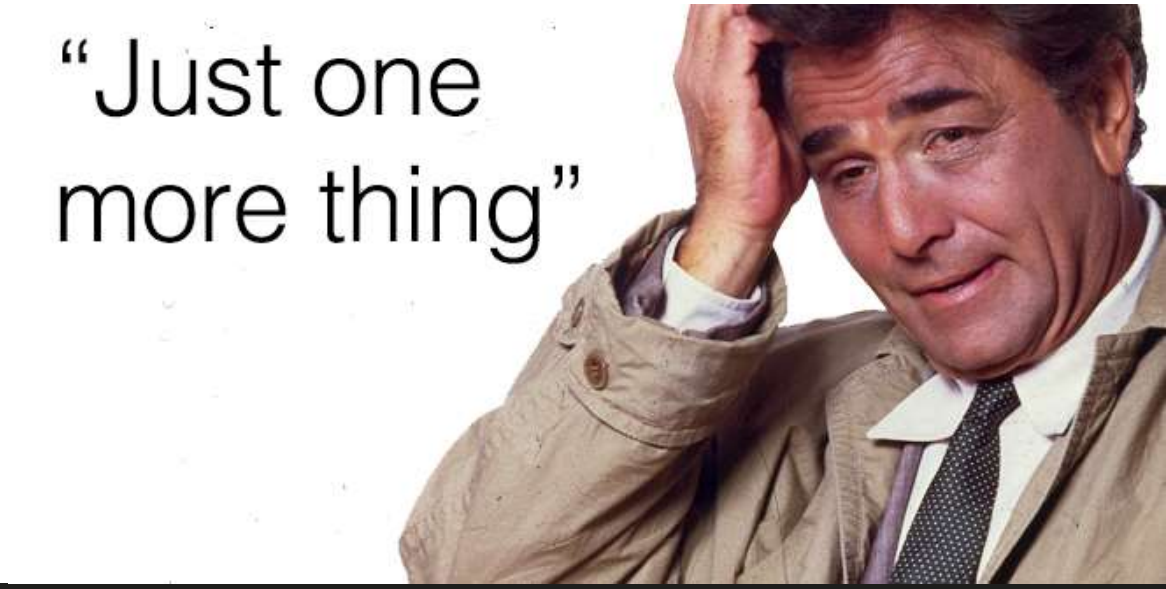
My Conclusion

It is very impressive what these projects have accomplished

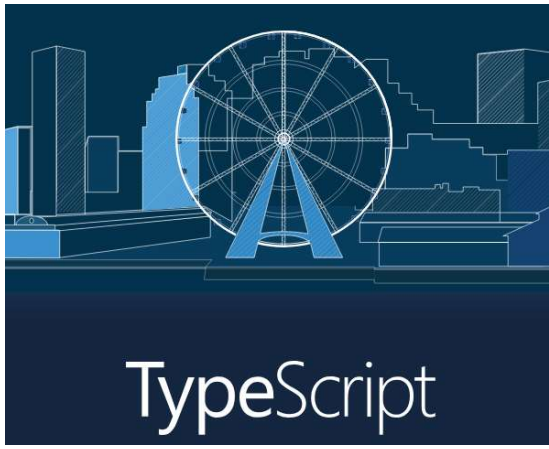
Some aspects of the development are succinct, safe and neat

If you are a functional fan & especially if doing F#, then this is pretty awesome

But…



My Conclusion



For my money, TypeScript is a better solution

Better balance (compromise) on safety, succinctness & productivity

- Gives me type safety (even if less expressive)
- Much easier interop with JavaScript (it's a superset)
- Easier consumption of existing JavaScript libraries
- Better tooling

I would stick with standard HTML/JSX for my view definitions

- Not because it is necessarily better for defining views
- Simply because the tooling, documentation and onboarding is easier

Angular has excellent TypeScript APIs