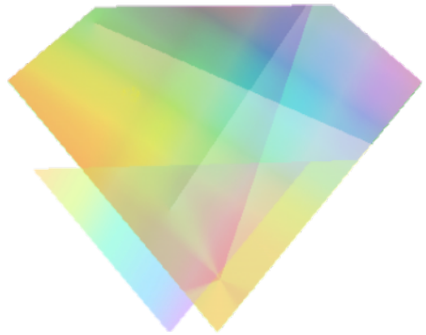
Everything has an end
Scala has two ...

Overview

- A load of languages compile to JS
- History of Scala JS
- Pro's and Contra's / Specifics of Scala JS
- Demo Time
- My use of Scala JS
- Discussion

Languages







One logo does not belong here ... which one?

History of Scala JS

Date	Version	What changed? (just a highlight)
2013/02/05,	**0.0**:	Initial Commit.
2013/11/29,	**0.1**:	Announcing Scala-js v0.1
2014/01/06,	**0.2**:	True Longs, already 7 contributors
2014/02/12,	**0.3**:	Futures, Promises
2014/03/13,	**0.4.0..4**:	#JsExport, many fixes
2014/06/13,	**0.5.0..6**:	Int/Char fastOptJS and fullOptJS
2014/12/01,	**0.6.0-M1**:	ClassCastException, js.native, Collections
2015/02/05,	**0.6.0**:	2-year anniversary, reached maturity

History of Scala JS

2015/03/03, **0.6.1..3**: java.nio.Buffer, Typed Arrays, BigInt, BigDec

2015/07/03, **0.6.4..5**: Pseudo type A|B, Extend native classes

2016/01/25, **0.6.6..10**: JS Tuples, js.ConstructorTag[C]

2016/07/27, **0.6.11..12**: Speed improvements

2016/10/17, **0.6.13..16**: @JSGlobalScope, @JSImport, CommonJS

2017/07/14, **0.6.17..22**: @JSImport with globalFallback, main(args)

2018/01/24, **0.6.22-25**: lazy vals

2018/11/29, **0.6.26**: ECMAScript modules

2017/07/03, **1.0.0-M1..5**: Global Scope, Inner classes

2018/10/24, **1.0.0-M6**: Latest version (for testing)

Benefits

- Use existing JS libraries (only wrap the good parts)
- Use existing Scala libraries (most work out-of-the-box)
- Profit from concise code with strong type checking
- Reuse and share your Scala code (and lib extensions)
- Wrap your own JS code

Heaven

- use 99% of the stuff under scala._
- use 95% if the stuff under java.lang._
- access the DOM, Canvas etc
- build/work with Eclipse, SBT
- pure Scala libs, Shapeless, Akka, ...

Hell

- JVM/Java dependent code
- java.lang.{Thread, Runtime}
- Reflection (so no calls on structural types / js.Any)

Purgatory

- Floats behave as doubles by default
- Runtime tests of Numbers are based on values
- Exceptions maybe 'undefined behaviour'
- Regular expressions
- Symbols (present, maybe dangerous)
- Enumerations

Type Correspondence

Scala Type	JS Type	Remarks
java.lang.String	string	
scala.Boolean	boolean	
scala.Char	opaque	
scala.Byte	number	-2^7 ... 2^7-1
scala.Short	number	-2^15 ... 2^15-1
scala.Int	number	-2^31 ... 2^31-1
scala.Long	opaque	Java Long does not fit
scala.Float	number	carefull: 1.4 is not a float!
scala.Double	number	NaN, -0.0 , ±∞
scala.Unit	undefined	
scala.Null	null	
subtypes of js.Object	corresponding JS type	
Scala (value) classes	partial	exported methods

Type Correspondence

Scala Type	JS Type	Remarks
scala.FunctionN	js.FunctionN	
mutable.Seq[T]	js.Array[T]	Not scala.Array[T]
mutable.Map[String, T]	js.Dictionary[T]	
Option[T]	js.UndefOr[T]	

Literal object construction

JS:
```
{foo: 42, bar: "foobar"}
```

Scala:
```
js.Dynamic.literal(foo = 42, bar = "foobar")
```

```
js.Dynamic.literal("foo" -> 42, "bar" -> "foobar")
```

Facades

Access a native JS objects with traits

```
@js.native
trait MyWindow extends js.Object
{ val document:    HTMLDocument = js.native
  var location:    String       = js.native
  def innerWidth:  Int = js.native
  def innerHeight: Int = js.native
  def alert(message: String): Unit = js.native
  def open(url: String, features: String = ""): Window = js.native
  def close(): Unit = js.native
  def `val`(): String = js.native
  @JSName("val")
  def value(): String = js.native }
```

Access a native JS objects by brackets

```
{ ...
  @JSBracketAccess
  def apply(index: Int): T = js.native
  @JSBracketAccess
  def update(index: Int, v: T): Unit = js.native
... }

js.Dynamic.global.foo     js.Dynamic.global.bar(0)
```

Intrusions

```
@JSExportTopLevel("foo.bar.HelloWorld")
object HelloWorld
{ @JSExport
  def sayHello() = { println("Hello world!") } }

@JSExportTopLevel("Foo")
class Bar(val x: Int)
{ override def toString(): String = s"Foo($x)" }

@JSExportAll
class Point(_x: Double, _y: Double)
{ val x: Double = _x
  var y: Double = _y
  def abs: Double = Math.sqrt(x*x + y*y)
  def sum: Double = x + y
  def sum_=(v: Double): Unit = y = v - x }

case class Point(
    @(JSExport @field) x: Double,
    @(JSExport @field) y: Double)
```

Time for action

- Stand alone application executed on Node JS
- Client (Scala JS) - Server (Scala JVM) Demo
- Publishing a cross (JS/JVM) library
- Larger IoT Application with wrapped native components

Finalisation

- Start with a skeleton
- Write Scala no JS

  JS: `["10", "10", "10", "10"].map(parseInt)` > `[10, NaN, 2, 3]`

  Scala: `List("10", "10", "10", "10").map(parseInt)` > `List(10, 10, 10, 10)`
- Discussion

Examples and code based on Scala.js Website and subpages thereof.