What is Scala.js

- Scala.js is a Scala -> Javascript compiler
- Write code in Scala, run in the browser
- *No more wallowing around in Javascript!*
  - No more fat-fingered typos making it to production
  - Good toolability/tool support
  - Strong, enforceable abstractions
  - Refactorability
object Example extends js.JSApp{
  def main() = {
    var x = 0
    while(x < 10) x += 3
    println(x)
    // 12
  }
}

ScalaJS.c.LExample$.prototype.main__V = (function() {
  var x = 0;
  while ((x < 10)) {
    x = ((x + 3) | 0)
  }
  ScalaJS.m.s_Predef().println__O__V(x)
  // 12
})();
Problems faced in Web Dev

- Our proprietary algorithm is $O(n \log(n))$ rather than $O(n \log(\log(n)))$
- The machine-learning team can’t reliably predict this user’s click behavior
- Nobody knows why this code works and we are afraid to touch it
Problems faced in Web Dev

- ✗ Our proprietary algorithm is $O(n \log(n))$ rather than $O(n \log(\log(n)))$
- ✗ The machine-learning team can’t reliably predict this user’s click behavior
- ✔ Nobody knows why this code works and we are afraid to touch it
11. “This” in JavaScript (tipling.com)

Vertical align anything with just 3 lines of CSS (zerosixthree.se)
Scala.js Today: the Tech

- Incremental compiles ~1s
- Dev executables ~ 1mb
- Deployed executables ~100-300kb, +5s
- Passes most of Scala’s own partest suite
- As fast as Raw Javascript
Scala.js Today: the Ecosystem

● Active Community
  ○ Mailing list ¾ as much traffic as scala-user

● >Dozen libraries available
  ○ Including Scalaz, Shapeless

● Mature platform
  ○ Incremental Compilation, IDE support, binary/backward-compatibility, ...
Live Demo

Client-Server Application
Cool Things

- DOM access is type-safe
- HTML generation is type-safe
- Ajax calls are type-safe
  - And Boilerplate-free!
- Hard to accidentally screw up
To Learn More...

- **Hands-on Scala.js, talk @ PNWScala**
  - Cool presentation I gave

- **Hands-on Scala.js E-book**
  - Lots of intro material on Scala.js

  - Main Website
Scala.js 14 Months Ago: Tech

- Dev turnaround: 30s
- Dev executables: ~20mb
- Deployable executables: 800kb, +100s
- No Tests
- >10x slower than Raw Javascript
Scala.js 14 Months Ago: Ecosystem

- No community
  - 2-3 people on the mailing list
- No libraries
- No tooling
Fancy Demo
Scala.js
Scala.js ??? Fancy Demo
Let’s talk about

The Tech

The Ecosystem
Let’s talk about

✗ The Tech

✓ The Ecosystem
Things We Need

✔ Web Server (Spray)

JavaScript APIs

HTML Generation
Things We Need

✔ Web Server (Spray)

JavaScript APIs

HTML Generation
JavaScript APIs

- Can access them dynamically
  - Annoying and unsafe

- Support for typed interop facades available
  - But no such facades written

- Tool to import typescript defs as facades
  - But it doesn't work all the time
Can access them dynamically

```javascript
import js.Dynamic.global

global.JSON.parse("[1, 2, 3]")

// [1, 2, 3]
```
Can access them dynamically

```javascript
import js.Dynamic.global

global.JSON.parse("[1, 2, 3]")
// [1, 2, 3]

global.JSON.parse("[1, 2, 3]")
// TypeError: undefined is not a function

global.JSN.parse("[1, 2, 3]")
// ReferenceError: JSN is not defined
```
Support for typed interop facades

object JSON extends js.Object {
    def parse(text: String): Dynamic = native
}

JSON.parse("[1, 2, 3]"
// [1, 2, 3]

JSON.parse("[1, 2, 3]"
// Compile error: value parse is not a member of object JSON
**TypeScript => Scala**

```typescript
interface StyleSheet {
    disabled: bool;
    ownerNode: Node;
    parentStyleSheet: StyleSheet;
    media: MediaList;
    type: string;
    title: string;
}
```

```scala
class StyleSheet extends js.Object {
    def disabled: Boolean = native
    def ownerNode: Node = native
    def parentStyleSheet: StyleSheet = native
    def media: MediaList = native
    def `type`: String = native
    def title: String = native
}
```
Doesn’t always work

- Buggy POC

- Scala & Typescript type-systems differ
  - e.g. Typescript has literal singleton types

- Solution: just fix it manually after
JavaScript APIs

• Batch import lib.d.ts from Typescript
• Manually fix up the things that don't work
• Publish compiled, *untested* facades to Maven Central as *scala-js-dom*

• Total work: ~4 hrs
Scala-Js-Dom

libraryDependencies +=

"org.scala-lang.modules.scalajs" %% "scalajs-dom" % "0.6"
Things We Need

✔ Web Server (Spray)

✔ JavaScript APIs (scala-js-dom)

HTML Generation
HTML Generation

- Games don't need HTML but websites do

- Options:
  - Cross-compile a Scala templating library
  - Write a wrapper for a JS templating library
  - Spend all day concatting strings
What didn't work

● Cross compiling Twirl, Scalate
  ○ Java dependencies

● Javascript templating libraries?
  ○ Won’t run on a Scala server

● Concatting strings
  ○ Just asking for XSS vulnerabilities
Cross compiling Scalate

<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>servlet-api</artifactId>
    <version>${servlet-api-version}</version>
</dependency>

<dependency>
    <groupId>com.sun.jersey</groupId>
    <artifactId>jersey-server</artifactId>
    <version>${jersey-version}</version>
</dependency>
Concatting Strings

document.innerHTML = "<h1>Hello " + name + "!</h1>"

...
Scalatags

● Existing, Pure Scala library

● No separate template files to load

● Zero dependencies
Scalatags

val frag = html(
  head(
    script(src="..."),
    script("alert('Hello')")
  ),
  body(
    div(
      h1(id="title", "My title"),
      p("Paragraph of text")
    )
  )
)
Scalatags

// Scala.js

libraryDependencies +=
  "com.scalatags" %% "scalatags" % "0.4.2"

// Scala-JVM

libraryDependencies +=
  "com.scalatags" %% "scalatags" % "0.4.2"
Things We Need

✔ Web Server (Spray)

✔ JavaScript APIs (scala-js-dom)

✔ HTML Generation (Scalatags)
What Next?

- We have HTML generation
- We have DOM APIs like XMLHttpRequest
- How do we make the Ajax calls typechecked?
Things We Need

✔ Web Server (Spray)

✔ JavaScript APIs (scala-js-dom)

✔ HTML Generation (Scalatags)

Type safe Ajax Routing
But Wait...

- Ajax calls involve Data
- Data needs to get sent between client & server
- Manually constructing {JSON, XML, CSV} blobs sucks
Things We Need

✔ HTML Generation (Scalatags)
✔ Web Server (Spray)
✔ JavaScript APIs (scala-js-dom)

Type safe Ajax Routing

Data Serialization Library
Requirements

- No Reflection

- Pure Scala
  - No Java
  - No Javascript

- Handles case classes
Things that don't Work

- Java serialization (Java)
- Kryo (Reflection)
- Play Json (Jackson/Java/Reflection)
- Spray Json (no case classes)
- Scala-Pickling (Reflection)

- ...
Basic Difficulty

- How to serialize case classes without Reflection?
- Need some way of breaking alpha equivalence
Basic Difficulty

- How to serialize case classes without Reflection?
- Need some way of breaking alpha equivalence
- Macros!
Writing my own: uPickle

- Basically Spray JSON with a macro for case classes
- ~1000 LOC
- Initially a pure-Scala (shared) JSON parser
  - Now JSON.parse in Scala.js, Jawn in Scala-jVM

- That was easy
Scala-Js-Dom

libraryDependencies += "com.lihaoyi" %% "upickle" % "0.2.5"
But wait...

- It cross compiles, but how do we know it works?

- For that matter, how do we know that Scalatags works?
Testing Options on Scala.js

- Blind Faith
- Manual Testing
- Jasmine
How Scalatags was tested

https://github.com/scala-js/scala-js/issues/96

...For **scalatags**, this basically involved copying and pasting the body of the unit tests into a separate project, **optimizeJSing**, and opening up my index.html in chrome to verify manually that it continues to do the right thing.

...
Things We Need

HTML Generation (Scalatags)
Web Server (Spray)
JavaScript APIs (scala-js-dom)
Type safe Ajax Routing
Data Serialization Library (uPickle)

Testing Framework
We need a Test Suite

- Manual testing libraries via C&Ping to example projects doesn't scale

- We already have one!
  - But it uses ScalaTest and only runs on Scala-JVM
What if...

We cross compile ScalaTest?

We cross-compile some subset of ScalaTest?

Find some other testing library?
### Popular Testing Libraries

<table>
<thead>
<tr>
<th>Library</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>specs2_2.10–2.3.11.jar</td>
<td>8.9 MB</td>
</tr>
<tr>
<td>scalatest_2.10–2.1.5.jar</td>
<td>7.2 MB</td>
</tr>
<tr>
<td>scalacheck_2.10–1.11.3.jar</td>
<td>885 KB</td>
</tr>
<tr>
<td>testng–6.8.8.jar</td>
<td>836 KB</td>
</tr>
<tr>
<td>junit–4.11.jar</td>
<td>245 KB</td>
</tr>
<tr>
<td>uTest_2.10–0.1.4.jar</td>
<td>165 KB</td>
</tr>
</tbody>
</table>

**Problem:** ScalaTest is huuuuge
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.TYPE)
public @interface Finders {
    String[] value();
}

Problem: ScalaTest uses Java sources
Problem: ScalaTest uses tons of Reflection

```scala
val fieldOption = objectWithProperty.getClass.getFields.find(isFieldToAccess)

val methodOption = objectWithProperty.getClass.getMethods.find(isMethodToInvoke)

val getMethodOption = objectWithProperty.getClass.getMethods.find(isGetMethodToInvoke)
```
What if...

We cross compile ScalaTest?

We cross-compile some subset of ScalaTest?

Find some other testing library?
It Works!

```scala
package org.scalatest
import scala.scalajs.test.JasmineTest

class FreeSpec extends JasmineTest {
  implicit class SuperString(s: String){
    def in(thunk: => Unit) = {
      it(s)(thunk)
    }
    def -(thunk: => Unit) = {
      describe(s)(thunk)
    }
  }
}

package scalatags
import org.scalatest._

class BasicTests extends FreeSpec{
  "basic tag creation" in {
    assert(a.toString === "<a/>")
    assert(html.toString === "<html/>)
    ...
  }
}
```
But...

● Super sketchy
  ○ What if the semantics differ?

● Only supports a very narrow subset of the API
  ○ Probably exactly the subset I want
  ○ ...but not the subset someone else would want
  ○ Not obvious what this subset is
What if...

We cross compile ScalaTest?

We cross-compile some subset of ScalaTest?

Find some other testing library?
Find some other testing library?

- Specs2 had much of the same problem
- Scalacheck is much more special purpose
- JUnit, test-ng, etc. are all out because Java
What if...

We cross compile ScalaTest?

We cross-compile some subset of ScalaTest?

Find some other testing library?

Writing my own
Writing my own:

µTest 0.2.4

uTest (pronounced micro-test) is a lightweight testing library for Scala. Its key features are:

- Less than 1000 lines of code
- A fancy set of macro-powered asserts
- A unique execution model
- Integration with SBT
- Cross compiles to ScalaJS
- Parallel testing
package mytests

object MyTestSuite extends TestSuite{
    val tests = TestSuite{
        'myTest - {
            val a = 1
            val b = 2
            assert(a == b)
        }
    }
}

Writing my own: uTest

- Basically ScalaTest’s Freespec + 2-3 asserts
- Written once and cross compiled
- Leaves out all the misc. things I don't need
- ~1000 LOC

- That was easy
uTest

libraryDependencies += "com.lihaoyi" %%% "utest" % "0.2.4"

libraryDependencies += "com.lihaoyi" % "utest" % "0.2.4"
scala-js-games

Roll ← scala-js-dom

Scala.js ← uTest

Scalatags

uPickle

Fancy Demo
Things We Need

✔ Web Server (Spray)
✔ JavaScript APIs (scala-js-dom)
✔ HTML Generation (Scalatags)

**Type safe Ajax Routing**

✔ Data Serialization Library (uPickle)
✔ Testing Framework (uTest)
What's Routing All About

- Call some method in some file with some arguments, return some value
What's Routing All About

• Call some method in some file with some arguments, return some value

• The rest of the features routing engines provide are purely cosmetic
What's Routing All About

- Call some method in some file with some arguments, return some value
- The rest of the features routing engines provide are purely cosmetic
- Don't need them for Ajax routes
Autowire: macro-based routing

trait Api{
  def endpoint(name: String, count: Int): Seq[String]
}

ajax[Api].endpoint("hello", 123).call(): Future[Seq[String]]

// becomes
ajax.makeRequest[Seq[String]](
  Seq("Api", "endpoint"),
  Map("name" -> ajax.write("hello"), "count" -> ajax.write(123))
)
Autowire: macro-based routing

```javascript
router.route[Api](cont)
// becomes
{
  case Request(Seq("Api", "endpoint"), args) =>
    router.write(cont.endpoint(
        router.read[String](args("name")),
        router.read[Int](args("count"))
    ))
...;
}
```
Autowire: macro-based routing

// Shared
trait Api{
    def endpoint(name: String, count: Int): Seq[String]
}

// Server
router.route[Api](new Api{
    def endpoint(name: String, count: Int) = ...
})

// Client
ajax[Api].endpoint("hello", 123).call()
One place to get it right

● Actual transport layer is left up to you to implement
  ○ `ajax.read`, `ajax.write`, `ajax.makeRequest`
  ○ `router.read`, `router.write`

● If you mess up, things will fail at runtime
  ○ But only need to get it right once
  ○ After that, all Ajax calls will be safe
  ○ read and write calls are trivial using uPickle
Autowire: Safety!

```scala
// Compile error: value endpoin is not a member of Controller

ajax[haoyi.Controller].endpoint("hello", "123").call()
// Compile error: type mismatch; found: String; required: Int

val x: Seq[String] =
// Compile error: type mismatch;
// found: Future[Seq[String]]
// required: Seq[String]
```
Autowire: Safety!

// OK
    doStuff(res: Seq[String])
}

// OK
val future1 = ajax[haoyi.Controller].endpoint("hello", 123).call()
val future2 = ajax[haoyi.Controller].endpoint("你好", 888).call()
for (res1 <- future1; res2 <- future2){
    doStuff(res1, res2)
}
Autowire

- Type-safe, boilerplate-free RPCs calls between Client & Server
- Returns a Future[T], so impossible to misuse
- Interestingly, *does not depend on uPickle*
  - Can be used on Scala-JVM with Kryo, pickling, etc.
- 435 LOC
Autowire

libraryDependencies += "com.lihaoyi" %% "autowire" % "0.2.3"

libraryDependencies += "com.lihaoyi" %% "autowire" % "0.2.3"
Things We Need

✔ Web Server (Spray)
✔ JavaScript APIs (scala-js-dom)
✔ HTML Generation (Scalatags)
✔ Type safe Ajax Routing (Autowire)
✔ Data Serialization Library (uPickle)
✔ Testing Framework (uTest)
Properties of the Scala.js Ecosystem

- Roughly breaks down into Javascript wrappers, and cross-built code

- No BS, minimal-dependency libraries
  - BS dependencies don’t exist in Scala.js
  - Servlets, Reflection, Classloaders, etc.

- No large frameworks (yet?)
Moral of the story?

- It takes quite a lot of effort to go from "working compiler" to "cool demo"
Moral of the story?

- It takes quite a lot of effort to go from "working compiler" to "cool demo"
- Writing things yourself ain't so bad
Moral of the story?

- It takes quite a lot of effort to go from "working compiler" to "cool demo"
- Writing things yourself ain't so bad
- If you are trapped on a desert island with nothing but a compiler, first thing to build is a testing framework
Bootstrapping the Scala.js Ecosystem

Questions?
To Learn More...

- **Hands-on Scala.js, talk @ PNWScala**
  ○ Cool presentation I gave

- **Hands-on Scala.js E-book**
  ○ Lots of intro material on Scala.js

- **http://www.scala-js.org/**
  ○ Main Website