Scala.js
Towards Utopia

Richard Dallaway, @d6y

underscore.io

Make Change Easier

Agenda

Part 1
Scala.js introduction

Part 2
Collaborative text editing with WOOT

Part 3
SBT, calling to and from Javascript…

Two Ideas

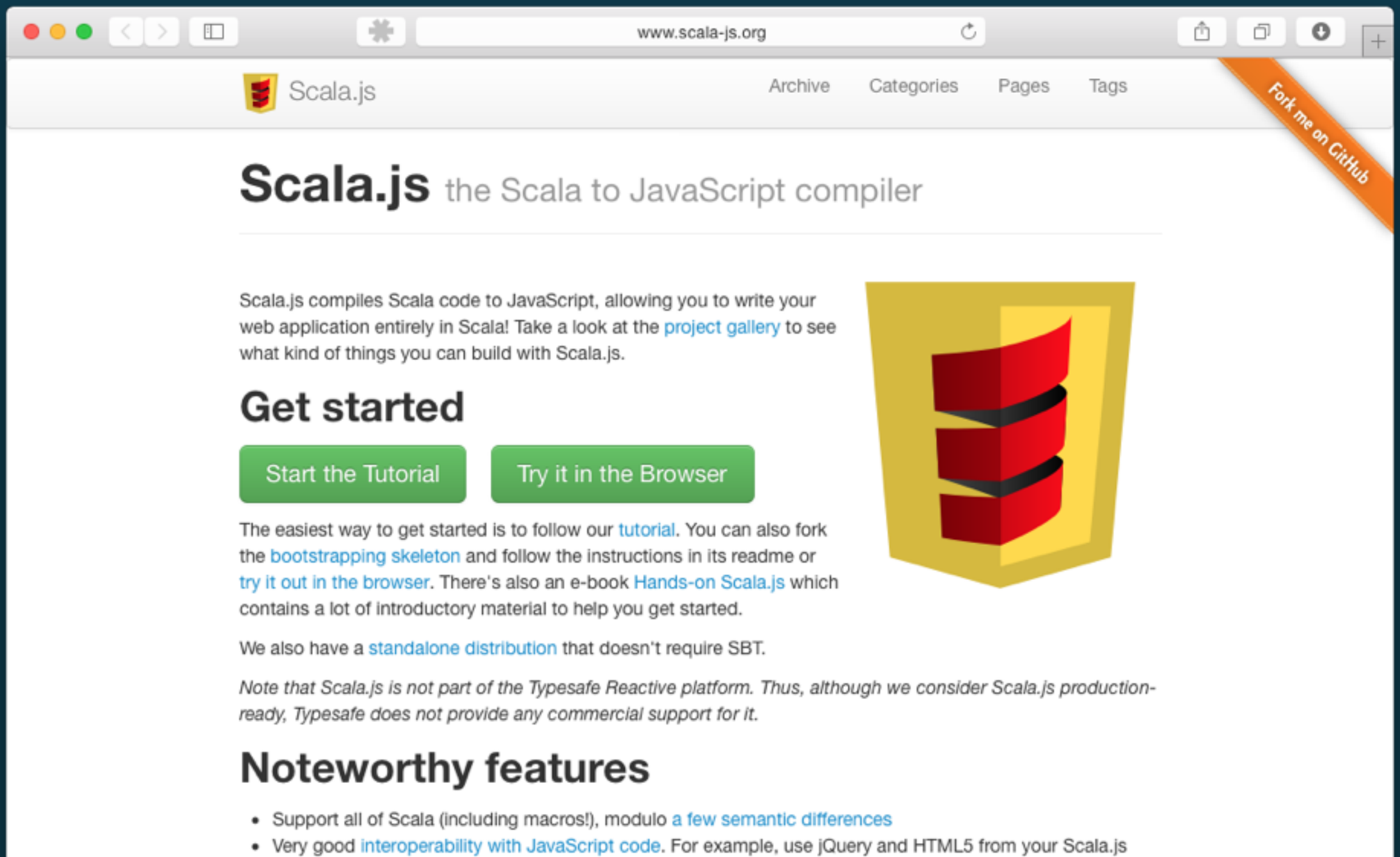
Gradually introduce Scala.js
to an existing code base

Distributed data types are
fun & a great fit with Scala.js

— Part 1 —

5 Minute Scala.js Intro

scala-js.org


www.scala-js.org
Scala.js
Archive
Categories
Pages
Tags
Fork me on GitHub
Scala.js the Scala to JavaScript compiler
Scala.js compiles Scala code to JavaScript, allowing you to write your web application entirely in Scala! Take a look at the project gallery to see what kind of things you can build with Scala.js.
Get started
Start the Tutorial
Try it in the Browser
The easiest way to get started is to follow our tutorial. You can also fork the bootstrapping skeleton and follow the instructions in its readme or try it out in the browser. There's also an e-book Hands-on Scala.js which contains a lot of introductory material to help you get started.
We also have a standalone distribution that doesn't require SBT.
Note that Scala.js is not part of the Typesafe Reactive platform. Thus, although we consider Scala.js production-ready, Typesafe does not provide any commercial support for it.
Noteworthy features
Support all of Scala (including macros!), modulo a few semantic differences
Very good interoperability with JavaScript code. For example, use jQuery and HTML5 from your Scala.js

Your Scala App Here

SBT Plugin

JavaScript

Std Lib

Libs, Macros…

Your Scala App Here

SBT Plugin

JavaScript

Std Lib

Libs, Macros…

Your Scala App Here

SBT Plugin

JavaScript

DOM

jQuery…

Yes, all of Scala

But not Java or reflection-based code

Yes, all of Scala

But not Java or reflection-based code

Ported	New	Typed JS Bindings
Scalaz	Scalatags	scalajs-dom
ScalaCheck	uTest	scalajs-jquery
shapeless	uPickle	scalajs-react
…	…	…

Five Basic Steps

addSbtPlugin("org.scala-js" % "sbt-scalajs" % "0.6.3")

Write Scala

@JSExport what you want expose

sbt> fastOptJs

Use resulting .js file

```scala
// Scala
def answer = Option(41).map(_ + 1)
```

```scala
// Scala
def answer = Option(41).map(_ + 1)
```

```javascript
// JavaScript
(function() {
  var o = Option().apply(41);
  if (o.isEmpty()) {
    var answer = None()
  } else {
    var arg1 = o.get();
    var answer =
      new Some().init(1 + arg1);
  };
  return answer
});
```

```scala
// Scala
def answer = Option(41).map(_ + 1)
```

```javascript
// JavaScript
(function() {
  var this$1 = $m_s_Option().apply__O__s_Option(41);
  if (this$1.isEmpty__Z()) {
    var answer = $m_s_None$()
  } else {
    var arg1 = this$1.get__O();
    var x$1 = $uI(arg1);
    var answer =
      new $c_s_Some().init___O(((1 + x$1) | 0))
  };
  return answer
});
```

Features

Fast Compiler

Optimized Output (this app: 295k)

Resulting JS is Fast

Great interop with JS

Community

Not a Framework

— Part 2 —

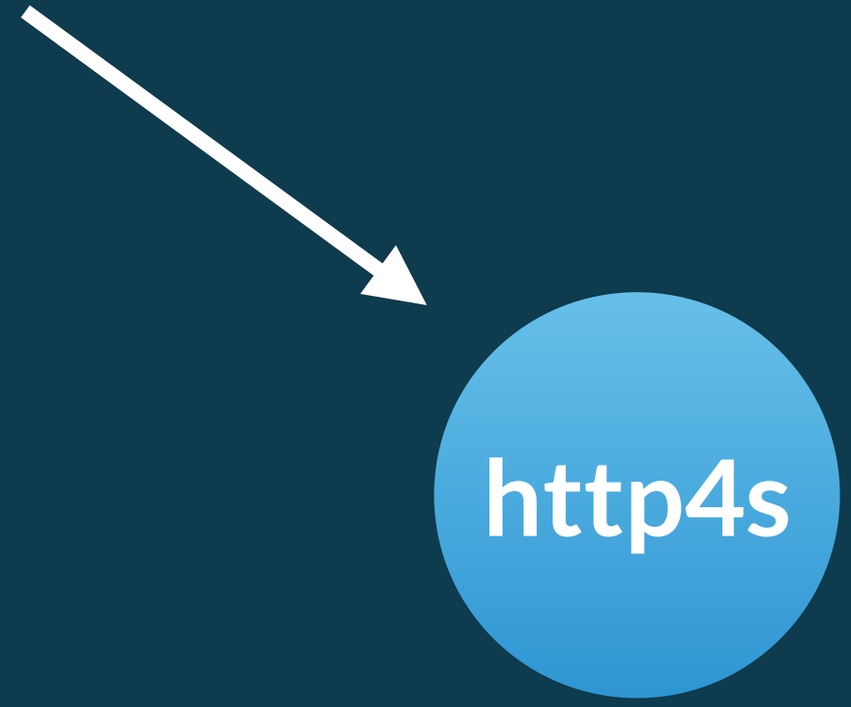
Collaborative App

What we want to build

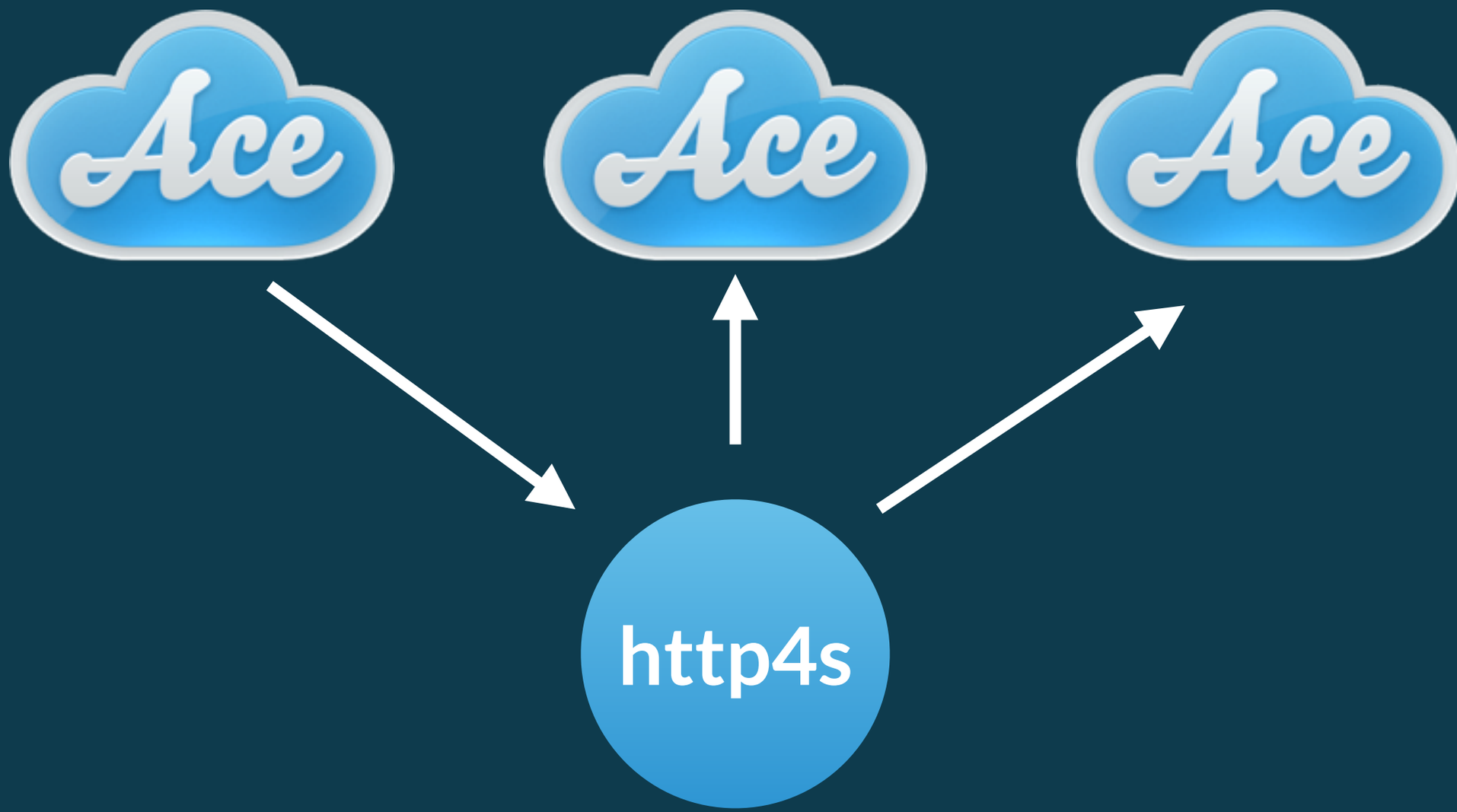
What we want to build

What we want to build

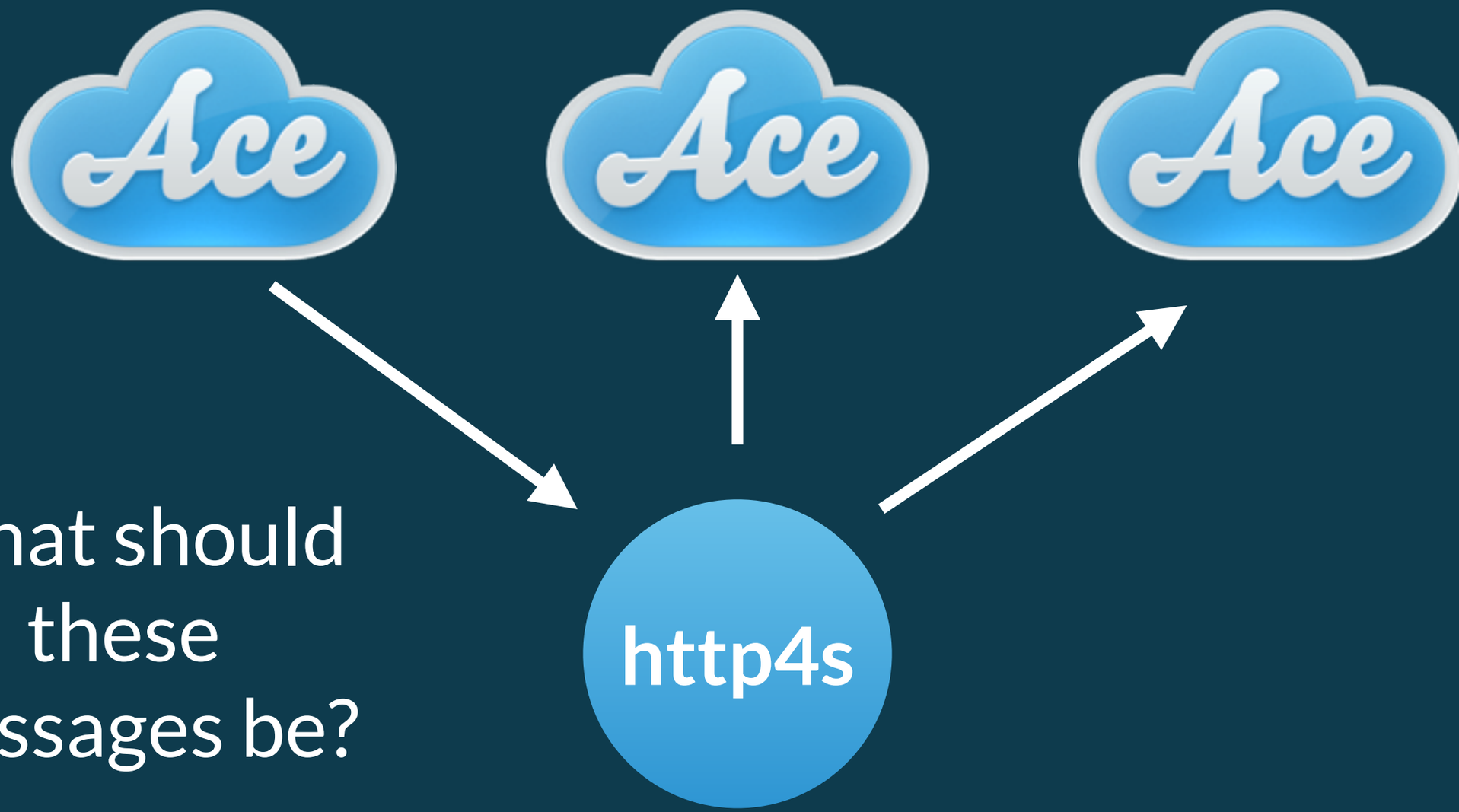
What we want to build



What we want to build



What we want to build



What should these messages be?

Don’t Do This

Alice’s View

C	A	T		
1	2	3		

Bob’s View

C	A	T		
1	2	3		

Don’t Do This

Alice’s View

C	A	T		
1	2	3		

Bob’s View

C	A	T		
1	2	3		

insert H @ 2

C	H	A	T	
1	2	3	4	

Don’t Do This

Alice’s View

C	A	T		
1	2	3		

insert H @ 2

C	H	A	T	
1	2	3	4	

Bob’s View

C	A	T		
1	2	3		

delete 3

C	A			
1	2			

Don’t Do This

Alice’s View

C	A	T		
1	2	3		

Bob’s View

C	A	T		
1	2	3		

insert H @ 2

C	H	A	T	
1	2	3	4	

delete 3

C	A			
1	2			

C	H	T		

C	H	A		

Commutative Replicated Data Type

WOOT

Alice's View

C	A	T		
1	2	3		

Bob's View

C	A	T		
1	2	3		

WOOT

Alice's View

C	A	T		
1	2	3		

Bob's View

C	A	T		
1	2	3		

C ≺ H ≺ A

C	H	A	T	
1	2	3	4	

WOOT

Alice's View

C	A	T		
1	2	3		

Bob's View

C	A	T		
1	2	3		

C ≺ H ≺ A

C	H	A	T	
1	2	3	4	

Delete T

C	A			
1	2			

WOOT

Alice's View

C	A	T		
1	2	3		

Bob's View

C	A	T		
1	2	3		

C < H < A

C	H	A	T	
1	2	3	4	

Delete T

C	A			
1	2			

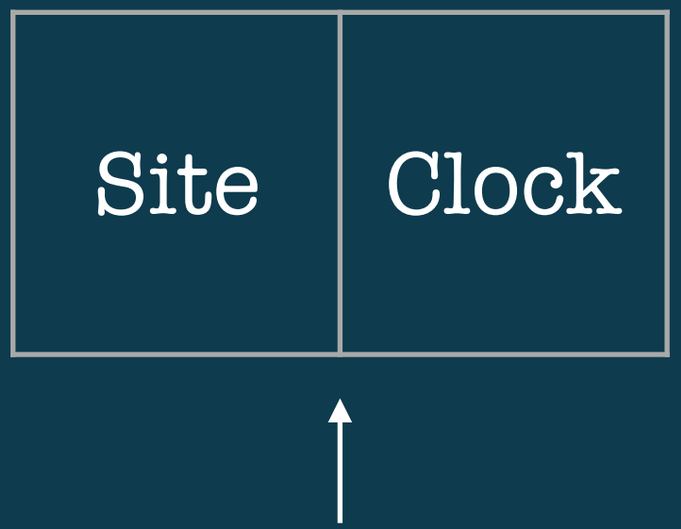
C	H	A		

C	H	A		

WChar

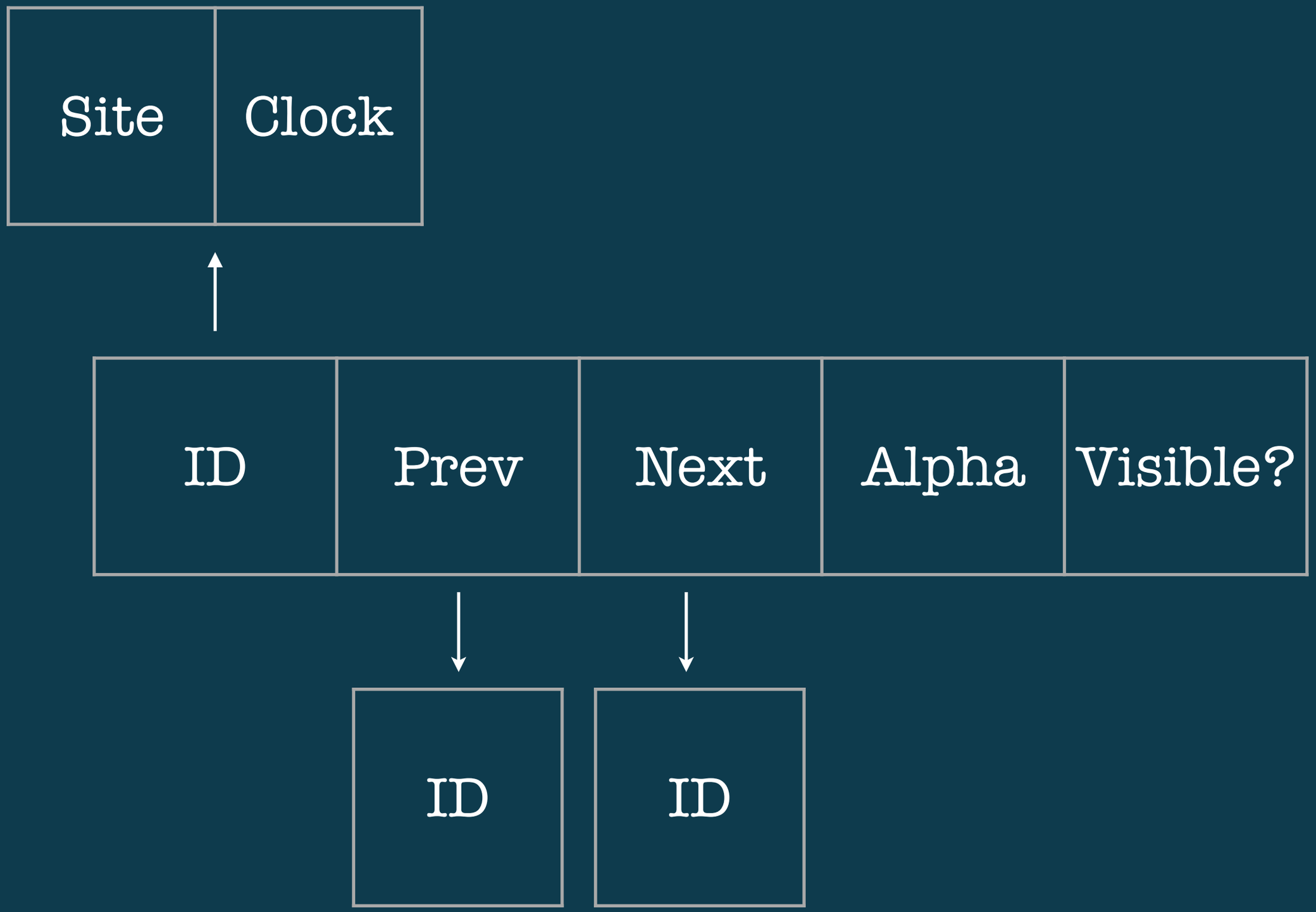
ID	Prev	Next	Alpha	Visible?

WChar

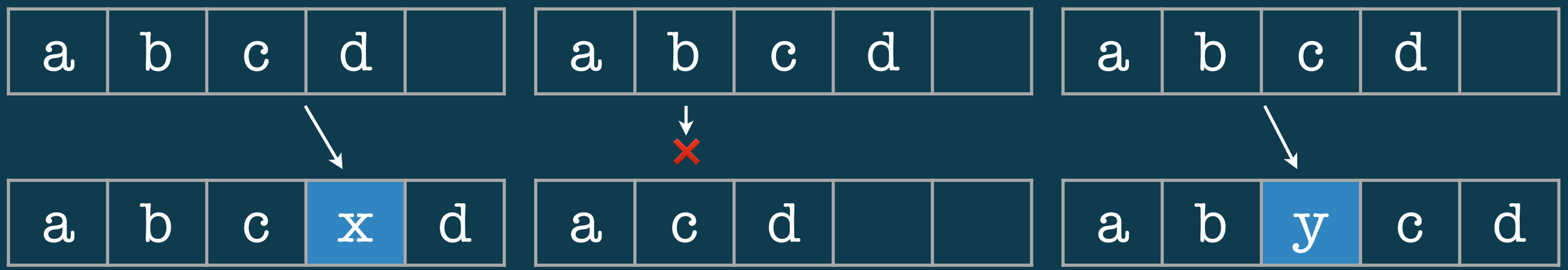


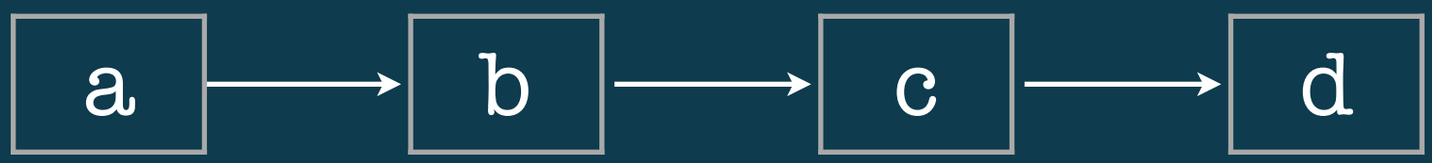
ID	Prev	Next	Alpha	Visible?

WChar

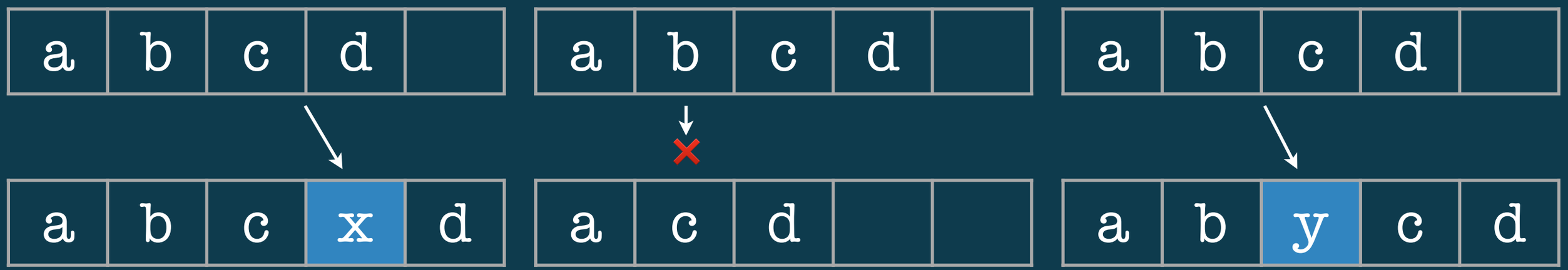


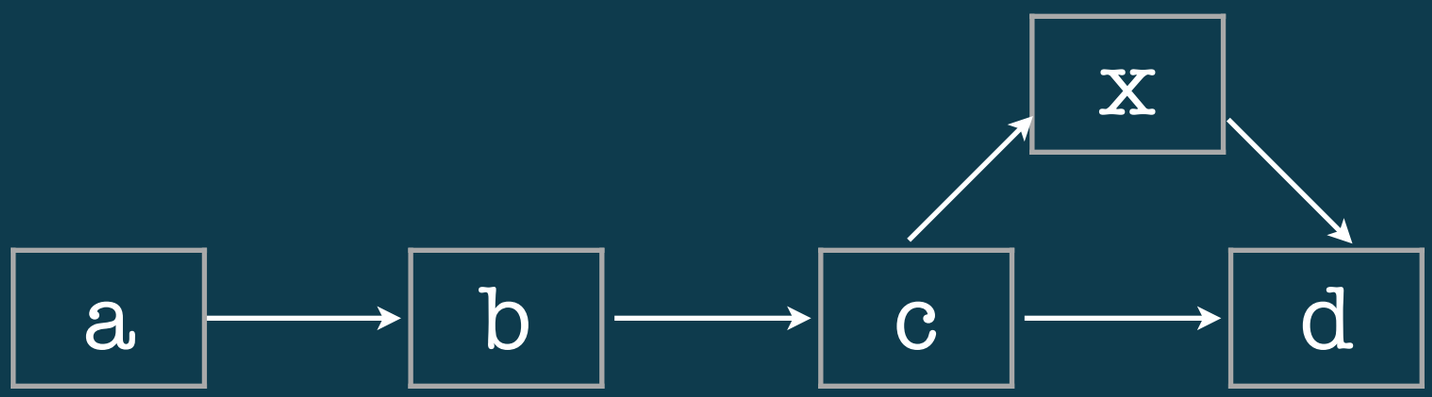
Algorithm



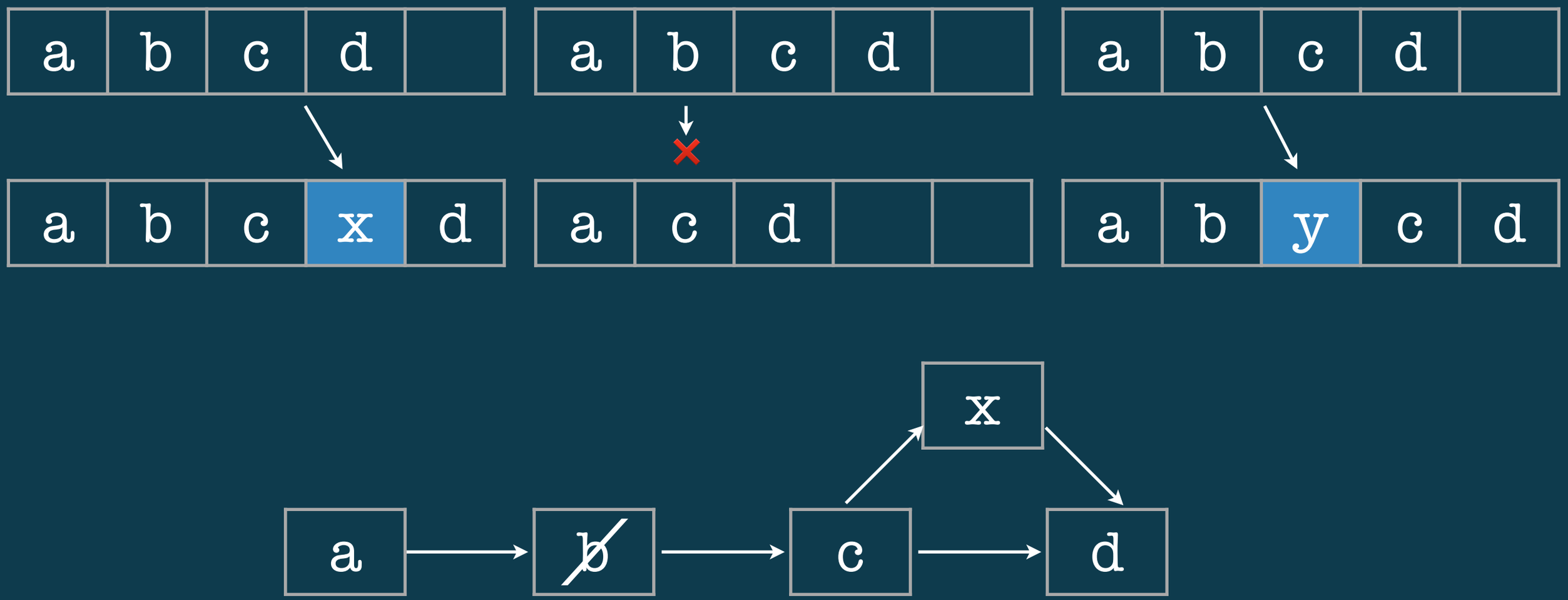


Algorithm

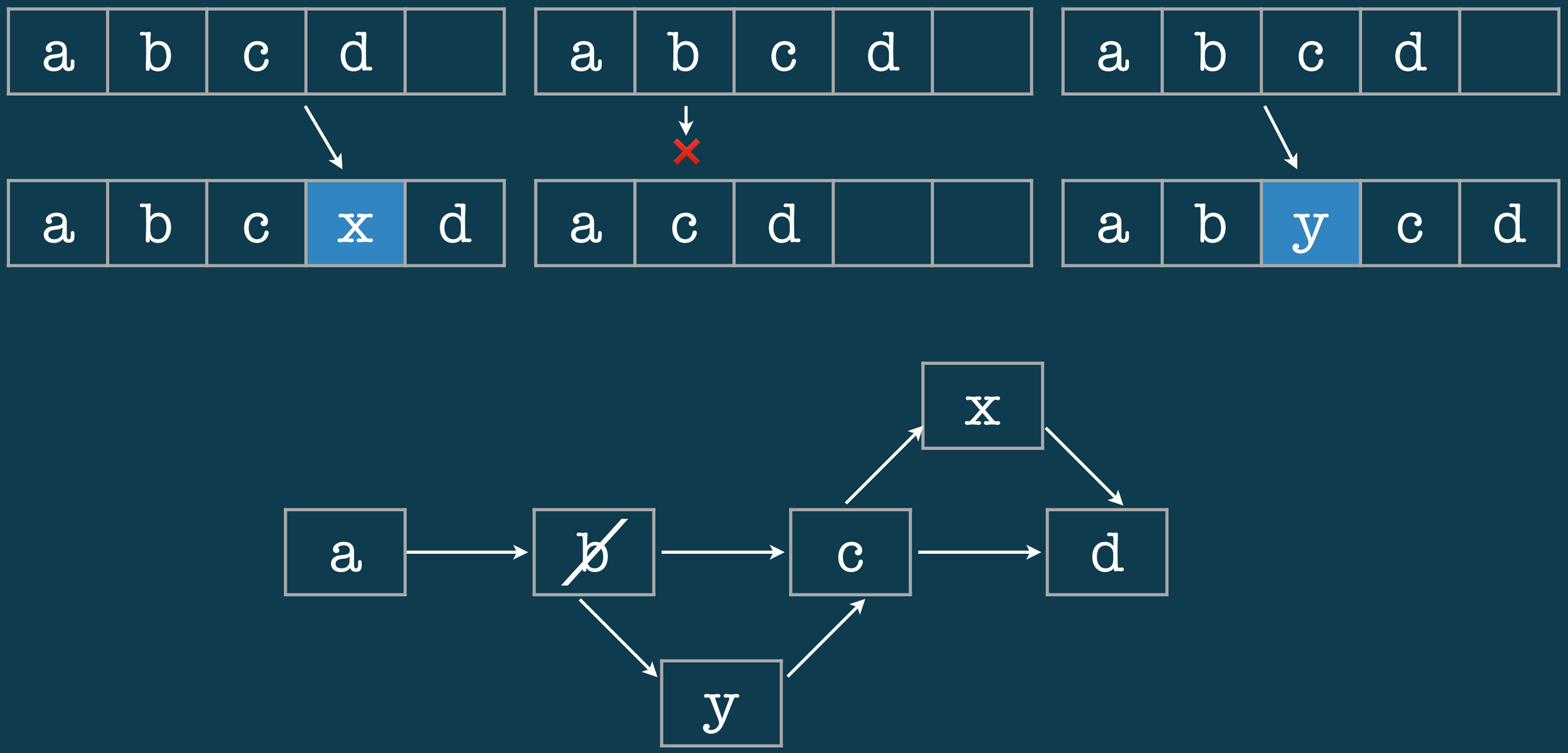




Algorithm



Algorithm



In JavaScript

Algorithm ~ 200 lines, 7.5k

Tests ~ 200 lines

+ require.js, underscore.js, jasmine

But now we have two problems

We’re going to look at…

Local Insert

Remote Ingest

We’re going to look at...

Local Insert

```
(WString,Char,Pos) => (WString,WChar)
```

Remote Ingest

We're going to look at...

Local Insert

```
(WString,Char,Pos) => (WString,WChar)
```

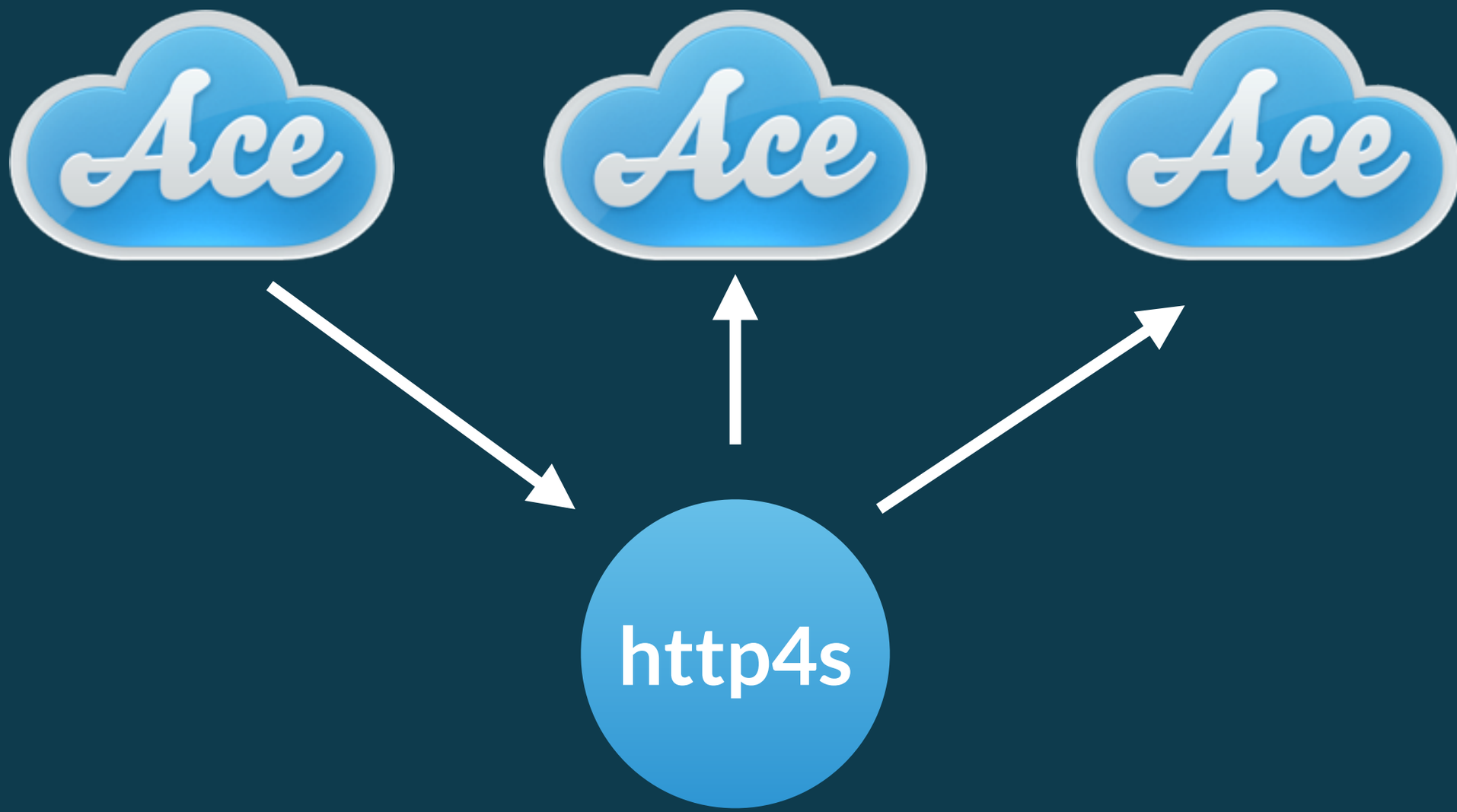
Remote Ingest

```
(WString,WChar) => WString
```

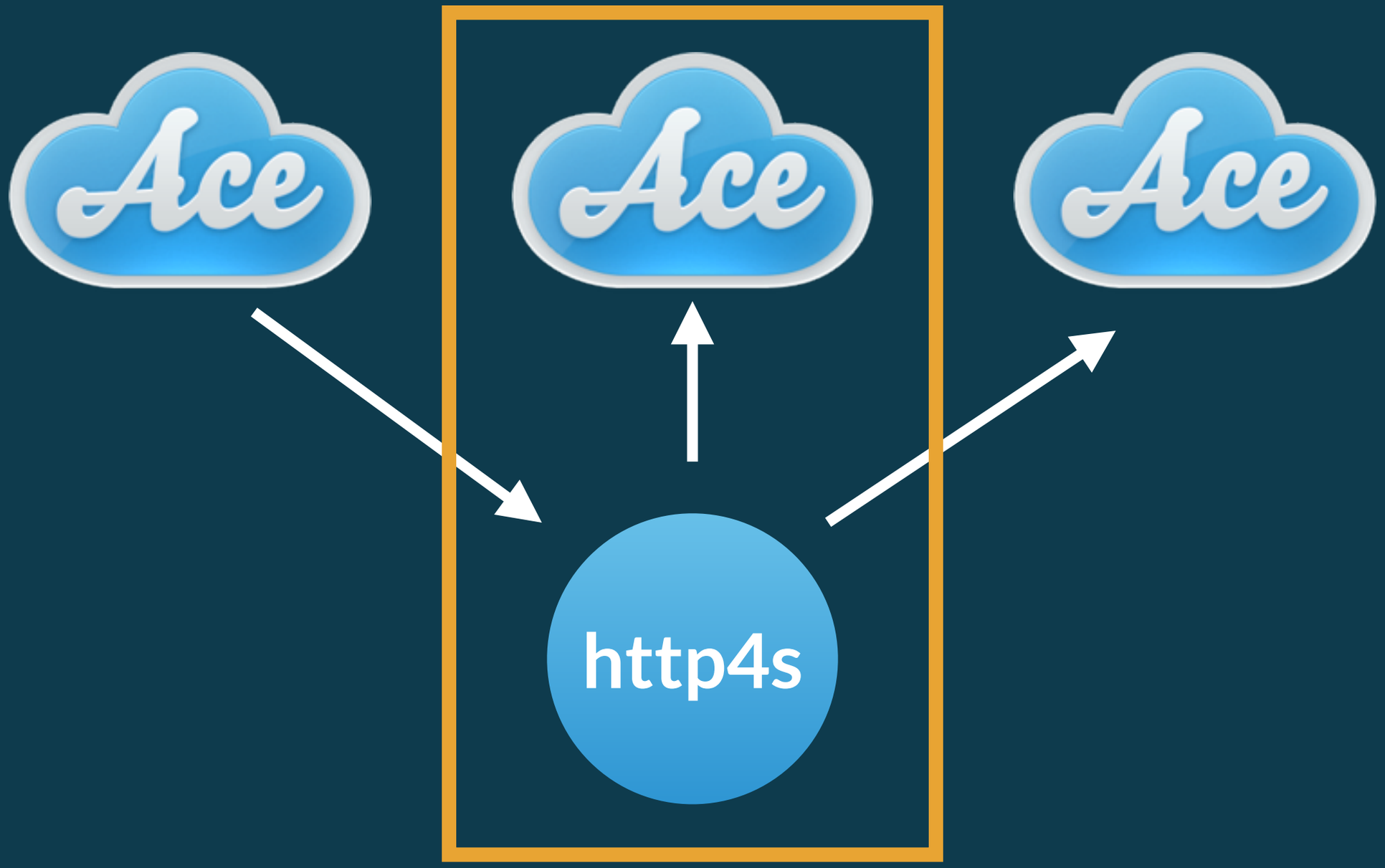
— Part 3 —

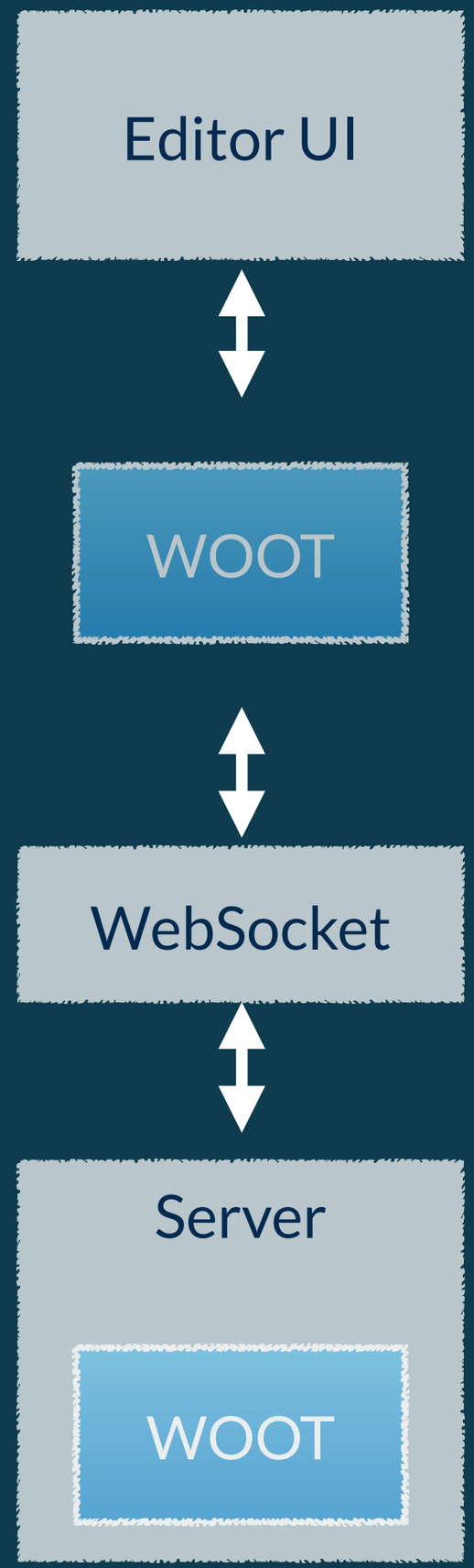
Migrating to Scala.js

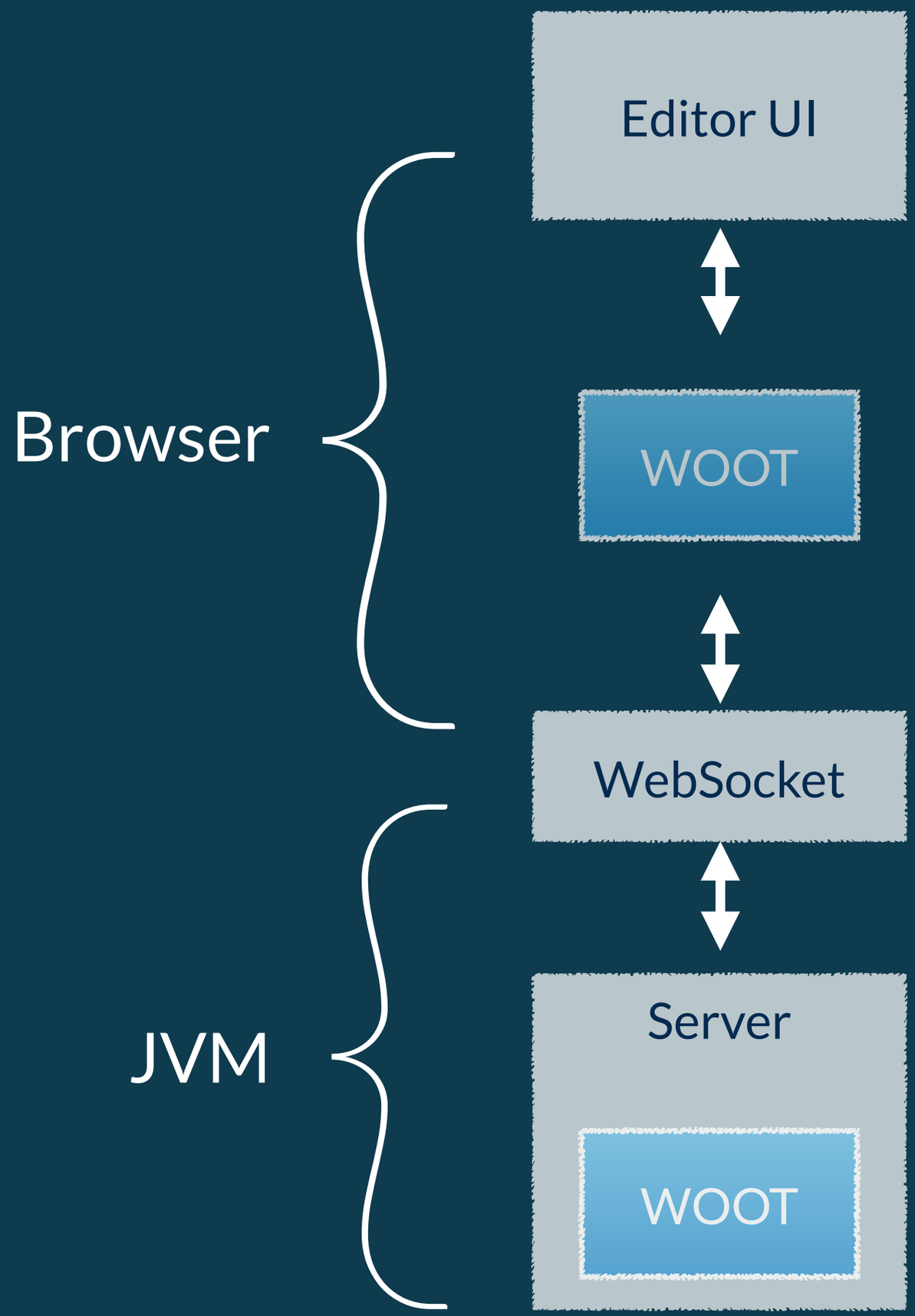
What we want to build

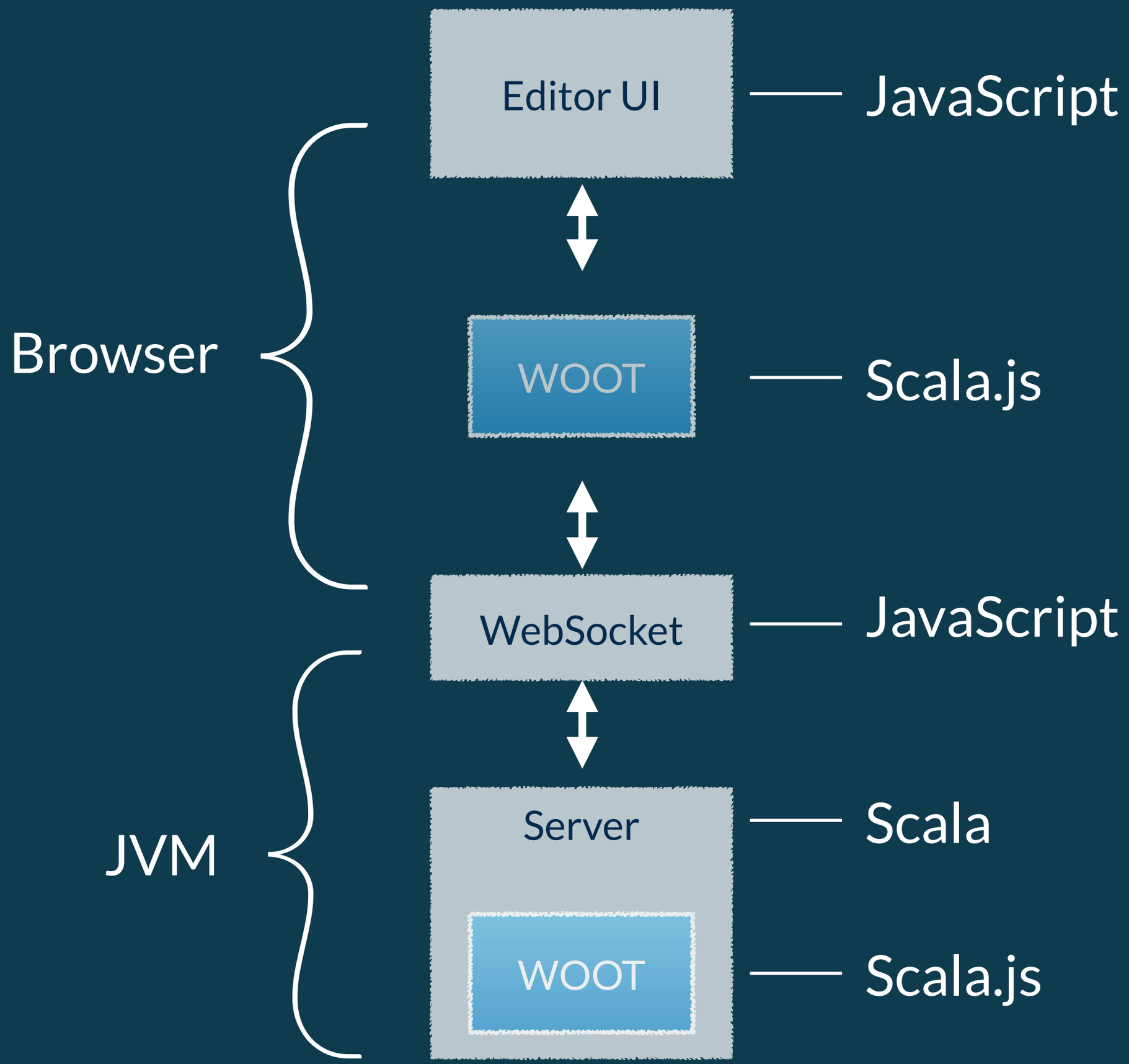


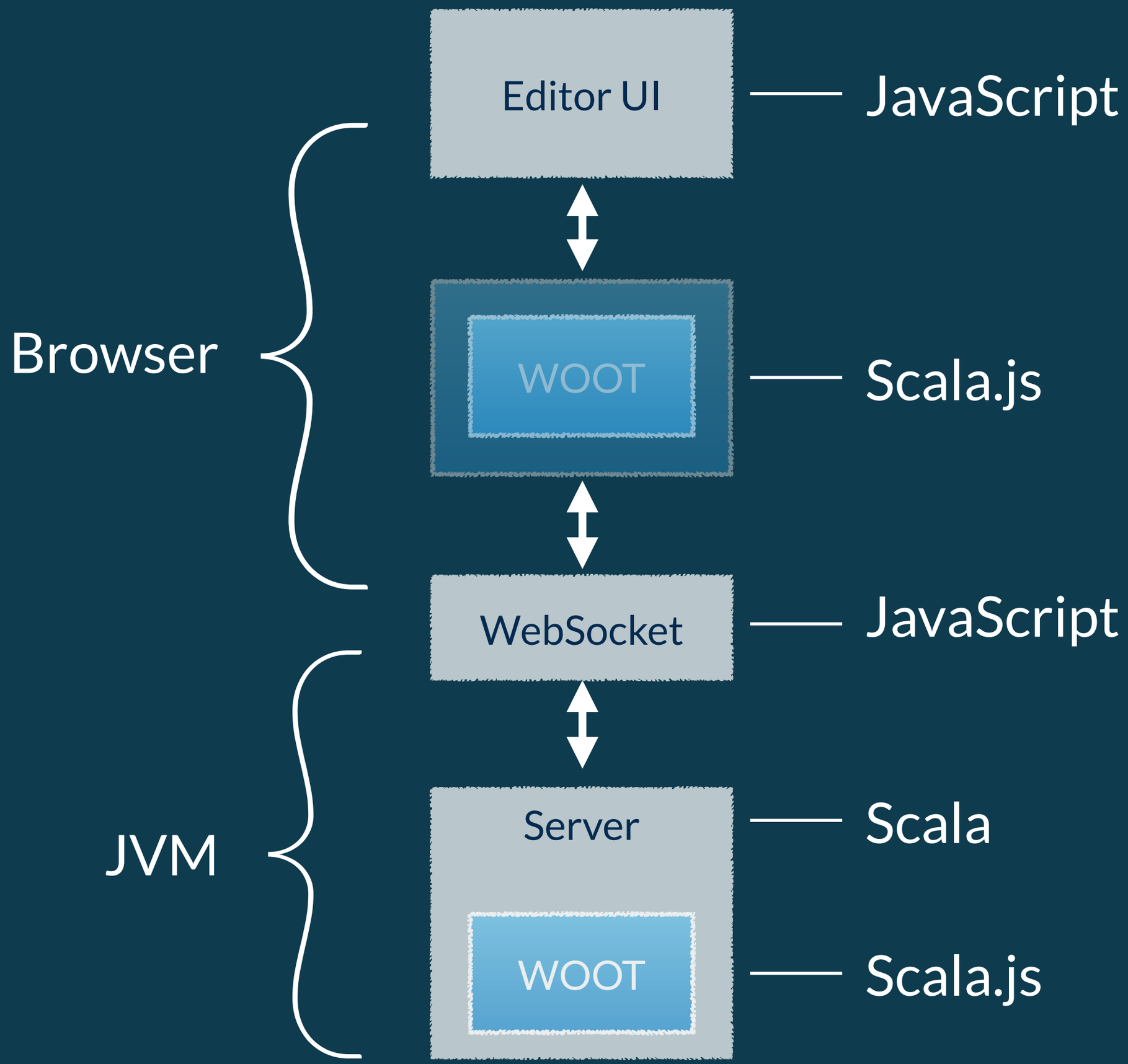
What we want to build




Editor UI
WOOT
WebSocket
Server
WOOT


Editor UI
Browser
WOOT
WebSocket
JVM
Server
WOOT


Editor UI
JavaScript
Browser
WOOT
Scala.js
WebSocket
JavaScript
JVM
Server
Scala
WOOT
Scala.js


Editor UI
JavaScript
Browser
WOOT
Scala.js
WebSocket
JavaScript
JVM
Server
Scala
WOOT
Scala.js

Editor UI

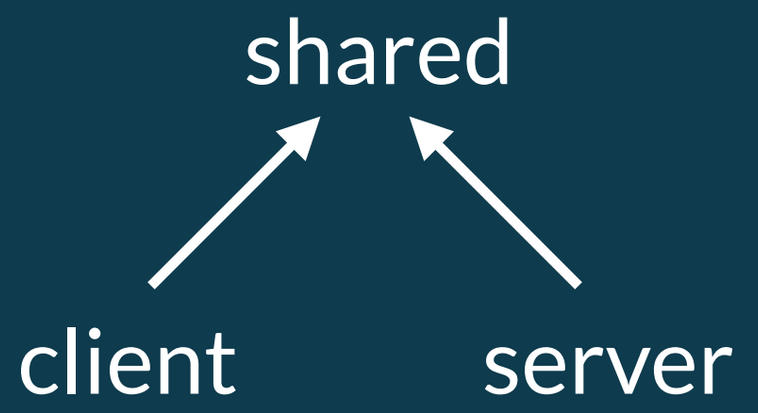
Client Wrapper

WOOT

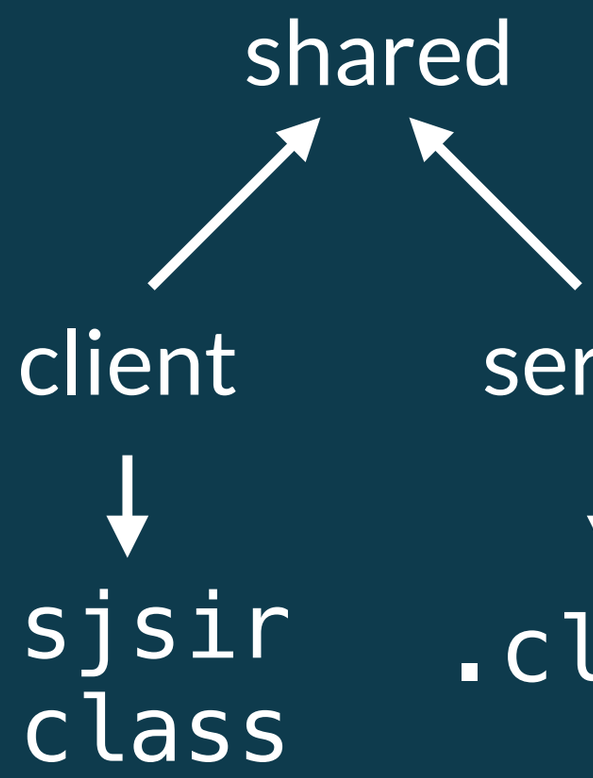
WebSocket

Server

build.sbt

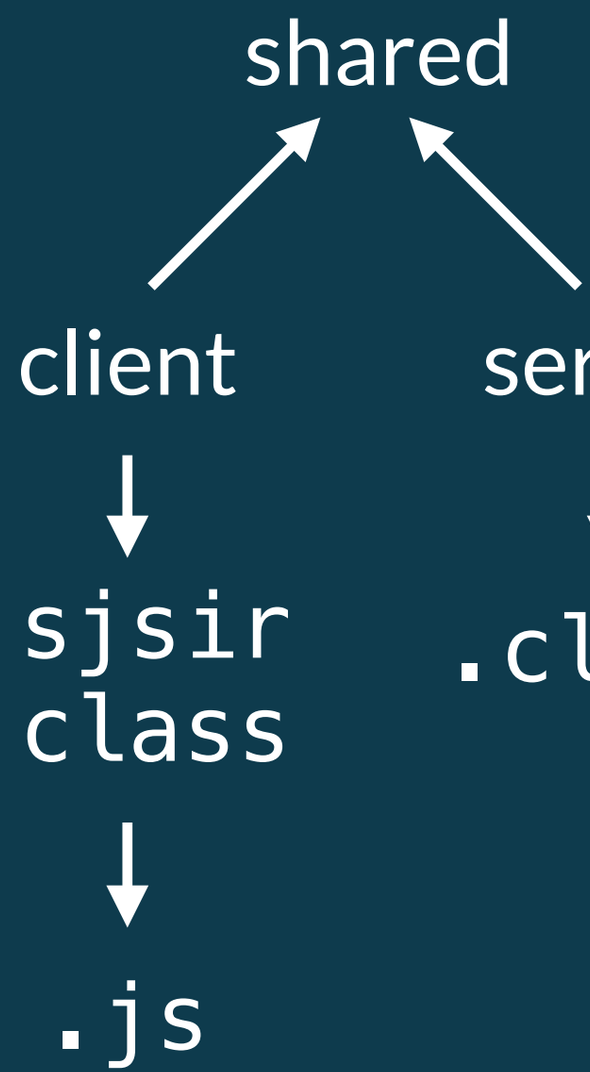


build.sbt




```
> compile
```

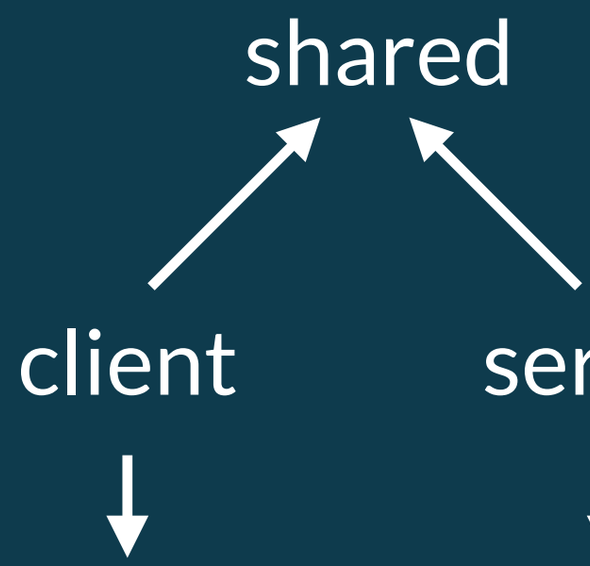
build.sbt




```
> compile

> fastOptJS
```

build.sbt




```
.sjsir
.class
```

```
.class
```

```
> compile
```

```
.class
```

```
> fastOptJS
```

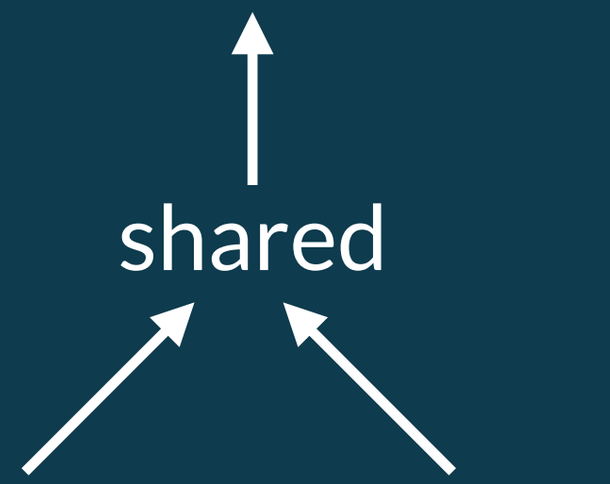
```
.js
```

```
> fullOptJS
```

```
.js
```

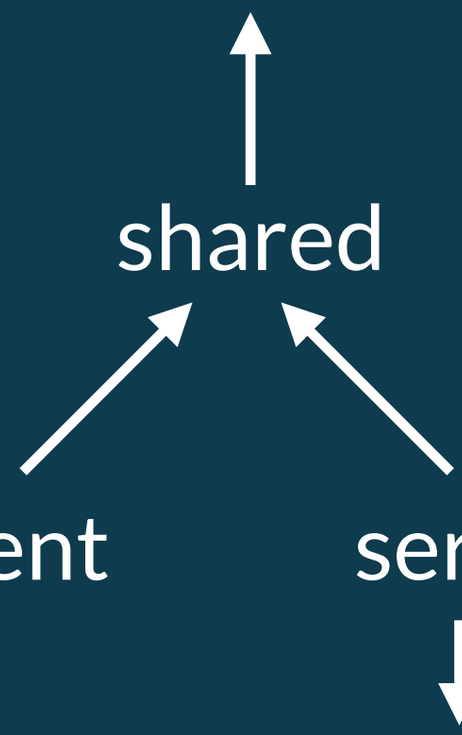
ScalaCheck

```
"org.scalacheck" %%% "scalacheck" % "1.12.2" % "test"
```



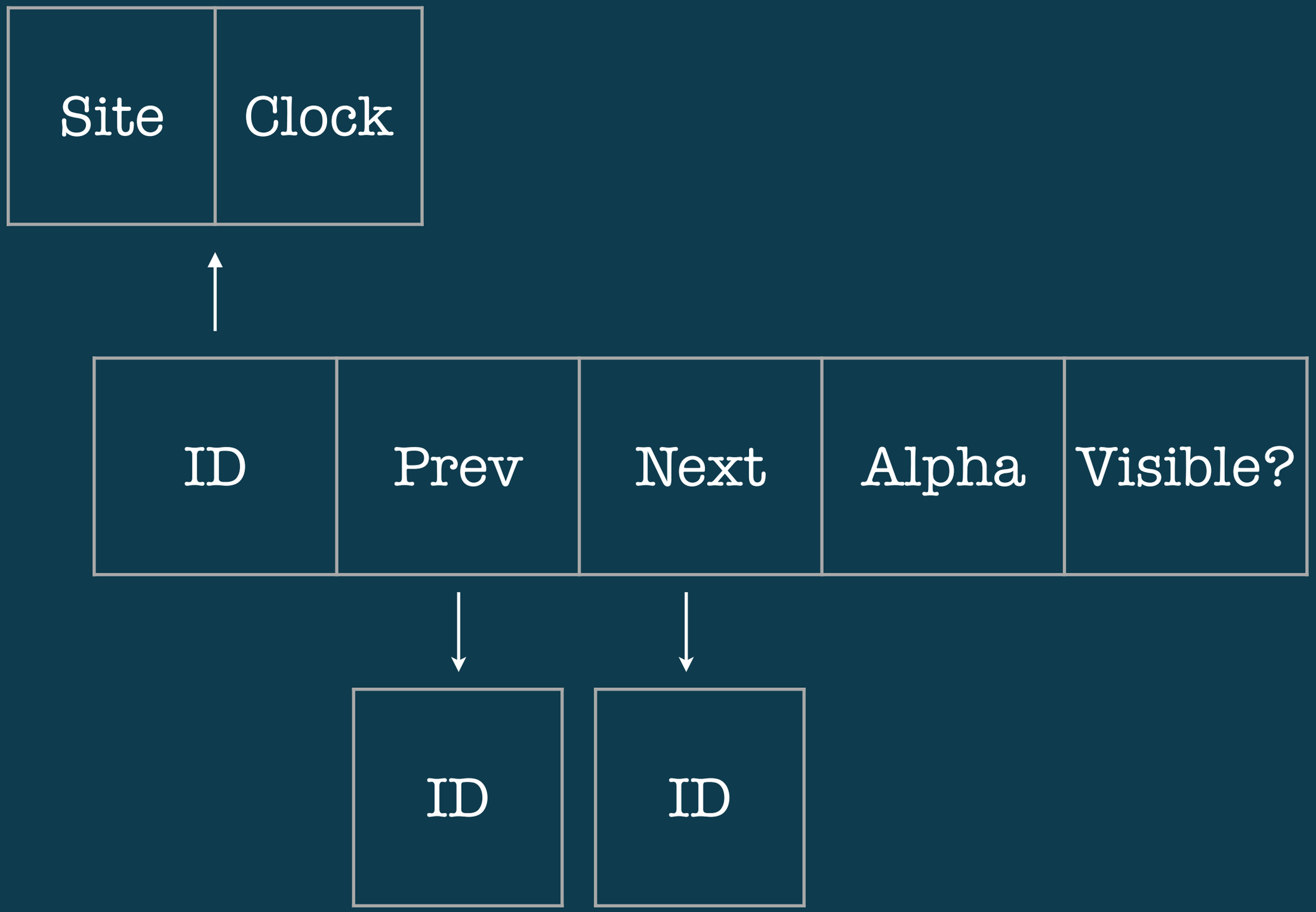
ScalaCheck

```
"org.scalacheck" %%% "scalacheck" % "1.12.2" % "test"
```




```
> test

[info] + Local insert preserves original text: OK, passed 250 tests.
[info] + Insert order is irrelevant: OK, passed 250 tests.
[info] + Inserting produces consistent text: OK, passed 250 tests.
[info] + Insert is idempotent: OK, passed 250 tests.
                              ⋮
```


Site
Clock
ID
Prev
Next
Alpha
Visible?
ID
ID

```scala
// Scala
import scala.scalajs.js
import js.annotation.JSExport

@JSExport
case class WChar(
          id: Id,
       alpha: Byte,
        prev: Id,
        next: Id,
  isVisible: Boolean = true)
```

```scala
// Scala
import scala.scalajs.js
import js.annotation.JSExport

@JSExport
case class WChar(
            id: Id,
         alpha: Byte,
          prev: Id,
          next: Id,
    isVisible: Boolean = true)

// HTML + JavaScript
<script src="client-fastopt.js">
<script>
  var char = new WChar(...)
</script>
```

OK, but...

```
case class WChar(
        id: Id,
    alpha: Byte,
      prev: Id,
      next: Id,
 isVisible: Boolean = true)
```

Should be Char?

Can’t this whole API be easier to use?

Semantics = Scala

There are differences

There are differences

JavaScript has no Char

toString differences

…not as many as you think

Wrapper Solution

```
package client

import js.annotation.JSExport
import woot.WString

@JSExport
class WootClient() {

  var doc = WString.empty()

  @JSExport
  def insert(s: String, pos: Int): Json = ???

  @JSExport
  def ingest(json: Json): Unit = ???
}
```

uPickle

```
sealed trait Operation {
  def wchar: WChar
}

case class InsertOp(override val wchar: WChar) extends Operation
case class DeleteOp(override val wchar: WChar) extends Operation
```

uPickle

```
sealed trait Operation {
  def wchar: WChar
}

case class InsertOp(override val wchar: WChar) extends Operation
case class DeleteOp(override val wchar: WChar) extends Operation
```

```
import upickle._

// Produce JSON
val op: Operation = ???
val json = write(op)
```

uPickle

```
sealed trait Operation {
  def wchar: WChar
}

case class InsertOp(override val wchar: WChar) extends Operation
case class DeleteOp(override val wchar: WChar) extends Operation
```

```
import upickle._

// Produce JSON
val op: Operation = ???
val json = write(op)

// Consume JSON
Try(read[Operation](json)).map( op => ...)
```

uPickle

```
sealed trait Operation {
  def wchar: WChar
}

case class InsertOp(override val wchar: WChar) extends Operation
case class DeleteOp(override val wchar: WChar) extends Operation
```

```
["woot.InsertOp", { "wchar": {
        "id": ["woot.CharId", {...}],
     "alpha": "*",
      "prev": ["woot.Beginning",{}],
      "next": ["woot.Ending",   {}]}
  }
]
```

```
@JSExport
class WootClient() {

  var doc = WString.empty()

  @JSExport
  def insert(s: String, pos: Int): Json = ???

  @JSExport
  def ingest(json: Json): Unit = ???
}
```

```
@JSExport
class WootClient() {

  var doc = WString.empty()

  @JSExport
  def insert(s: String, pos: Int): Json = {
    val (op, wstring) = doc.insert(s.head, pos)
    doc = wstring
    write(op)
  }

  @JSExport
  def ingest(json: Json): Unit = ???
}
```

Effecting the DOM

```
import org.scalajs.dom
val element = dom.document.getElementById("editor")
```

Effecting the DOM

```
import org.scalajs.dom
val element = dom.document.getElementById("editor")


// DANGER! Run away.
// Unsafe access to Ace's API
val ace = js.Dynamic.global.ace

ace.edit("editor")
    .getSession()
    .getDocument()
    .setValue("We have control")

// Risk JS Uncaught type error
```

Effecting the DOM

```
// JavaScript
var updateEditor = function(s, isVisible) {
  var delta = convertWootToAceCoordinates(…);
  editor.getSession()
        .getDocument().applyDeltas([delta]);
}
```

Effecting the DOM

```
// JavaScript
var updateEditor = function(s, isVisible) {
  var delta = convertWootToAceCoordinates(…);
  editor.getSession()
        .getDocument().applyDeltas([delta]);
}

jQuery(document).ready(function() {
  client = new client.WootClient(updateEditor);
});
```

```
@JSExport
class WootClient() {

  var doc = WString.empty()

  @JSExport
  def ingest(json: Json): Unit = ???
}
```

```
@JSExport
class WootClient(f: js.Function2[String,Boolean,Unit]) {

  var doc = WString.empty()

  @JSExport
  def ingest(json: Json): Unit = ???
```

```
}
```

```
@JSExport
class WootClient(f: js.Function2[String,Boolean,Unit]) {

  // JavaScript
  var updateEditor = function(s, isVisible) {…

}
```

```
@JSExport
class WootClient(f: js.Function2[String,Boolean,Unit]) {

  var doc = WString.empty()

  @JSExport
  def ingest(json: Json): Unit =
    Try(read[Operation](json)).foreach(applyOperation)

  def applyOperation(op: Operation): Unit = ???

}
```

```
@JSExport
class WootClient(f: js.Function2[String,Boolean,Unit]) {

  var doc = WString.empty()

  @JSExport
  def ingest(json: Json): Unit =
    Try(read[Operation](json)).foreach(applyOperation)

  def applyOperation(op: Operation): Unit = {
    val (ops, wstring) = doc.integrate(op)

    // Become the updated document:
    doc = wstring

    // Side effects:
    ops.foreach {
      case InsertOp(ch) => f(ch.alpha.toString, true)
      case DeleteOp(ch) => f(ch.alpha.toString, false)
    }
  }
}
```

```scala
def integrate(op: Operation): (Vector[Operation], WString) = op match {
  // - Don't insert the same ID twice:
  case InsertOp(c,_) if chars.exists(_.id == c.id) => (Vector.empty, this)

  // - Insert can go ahead if the next & prev exist:
  case InsertOp(c,_) if canIntegrate(op) =>
    val (ops, doc) = integrate(c, c.prev, c.next).dequeue()
    (op +: ops, doc)

  // - We can delete any char that exists:
  case DeleteOp(c,_) if canIntegrate(op) => (Vector(op), hide(c))

  // - Anything else goes onto the queue for another time:
  case _                                 => (Vector.empty, enqueue(op))
 }
```

```scala
@scala.annotation.tailrec
private def integrate(c: WChar, before: Id, after: Id): WString = {
 // Looking at all the characters between the previous and next positions:
 subseq(before, after) match {
 // - when where's no option about where to insert, perform the insert
 case Vector() => ins(c, indexOf(after))

 // - when there's a choice, locate an insert point based on `Id.<`
 case search: Vector[WChar] =>
   val L: Vector[Id] = before +: trim(search).map(_.id) :+ after
   val i = math.max(1, math.min(L.length-1, L.takeWhile(_ < c.id).length))
   integrate(c, L(i-1), L(i))
}
```

What we've Seen

Multi-project build ✓

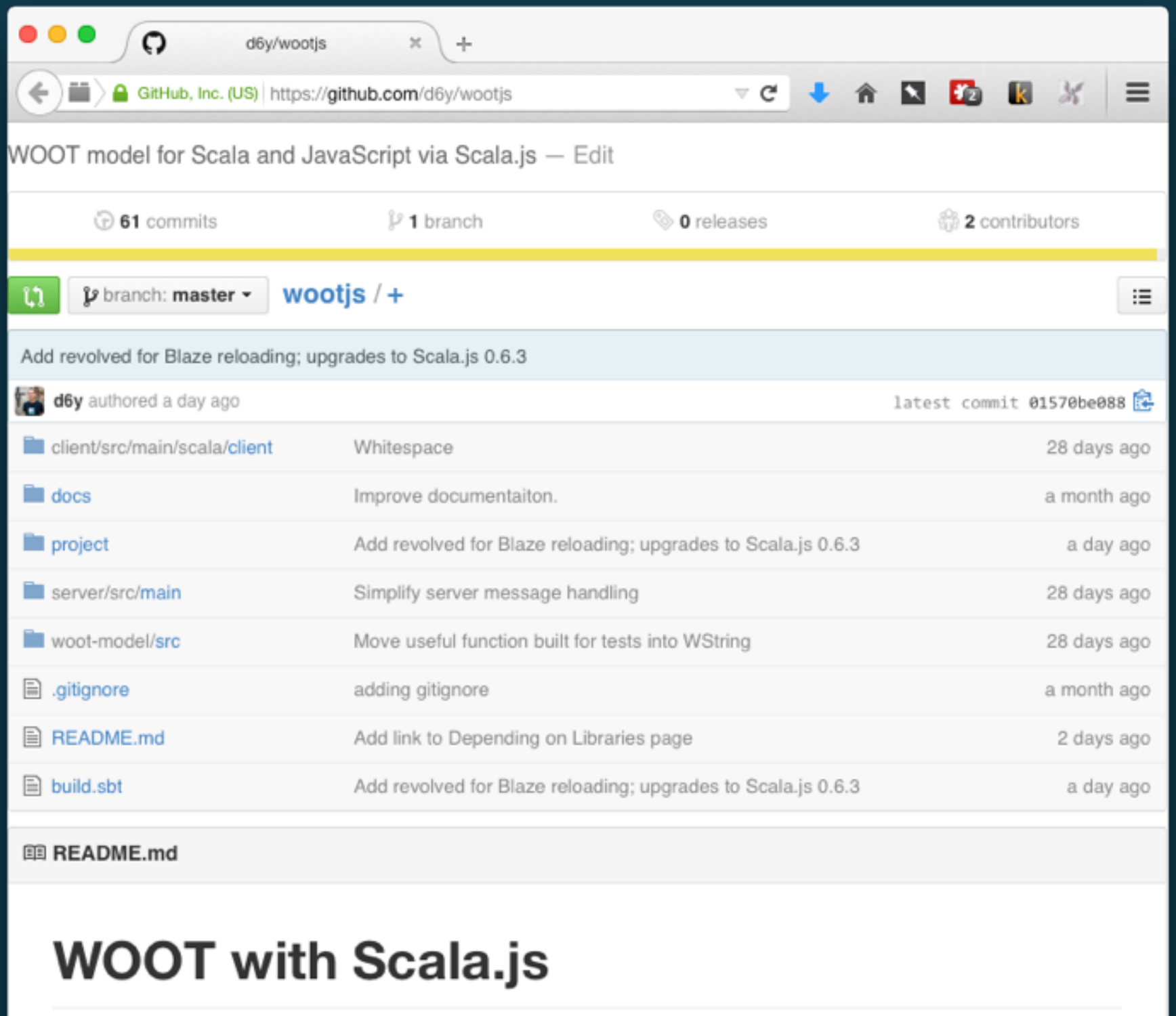
Wrote Scala, ran it both places ✓

Great interop (call JS, be called by JS) ✓

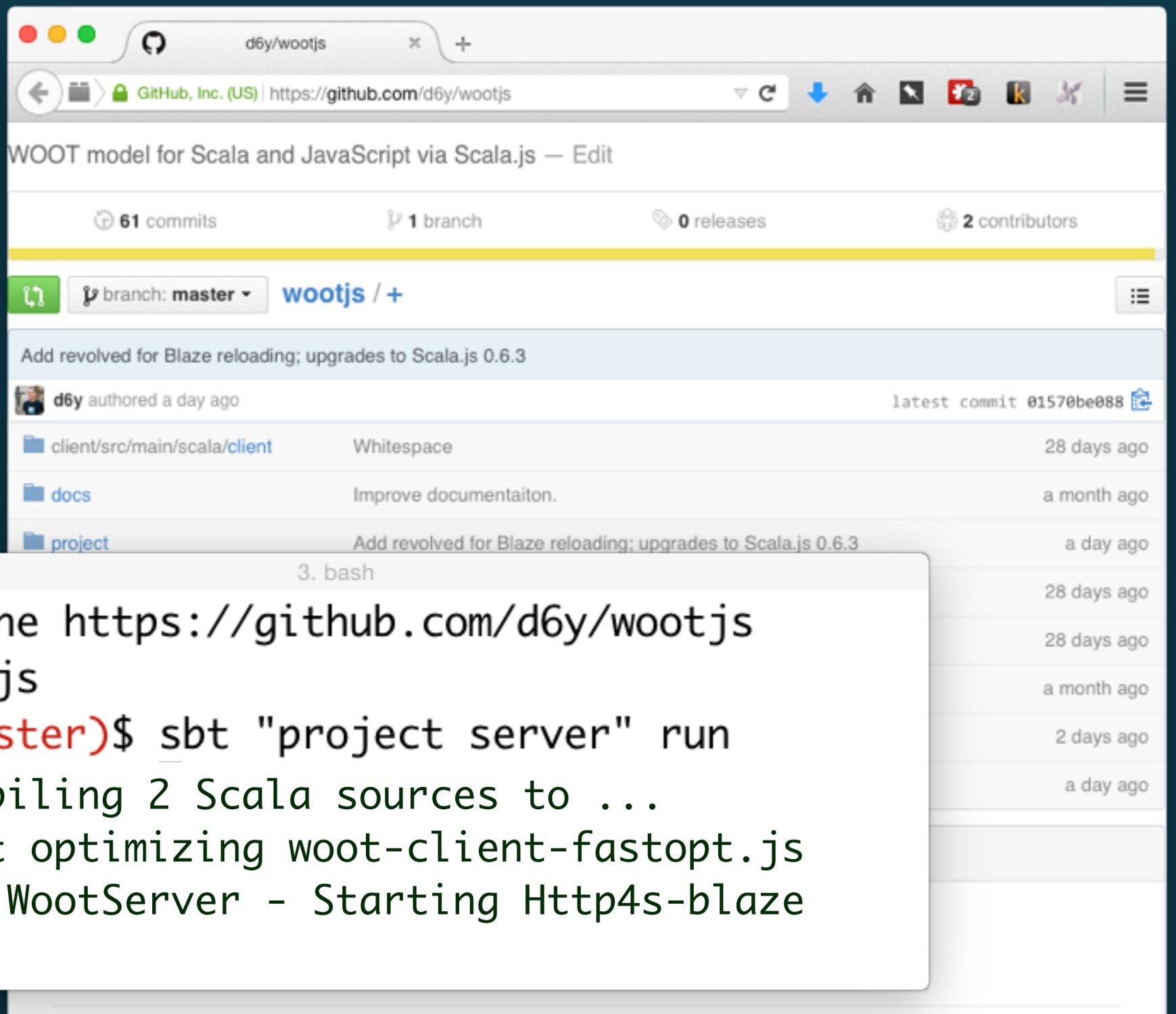
Dirty dirty dynamic calls ✓

Used cross-compiled libraries ✓

github.com/d6y/wootjs



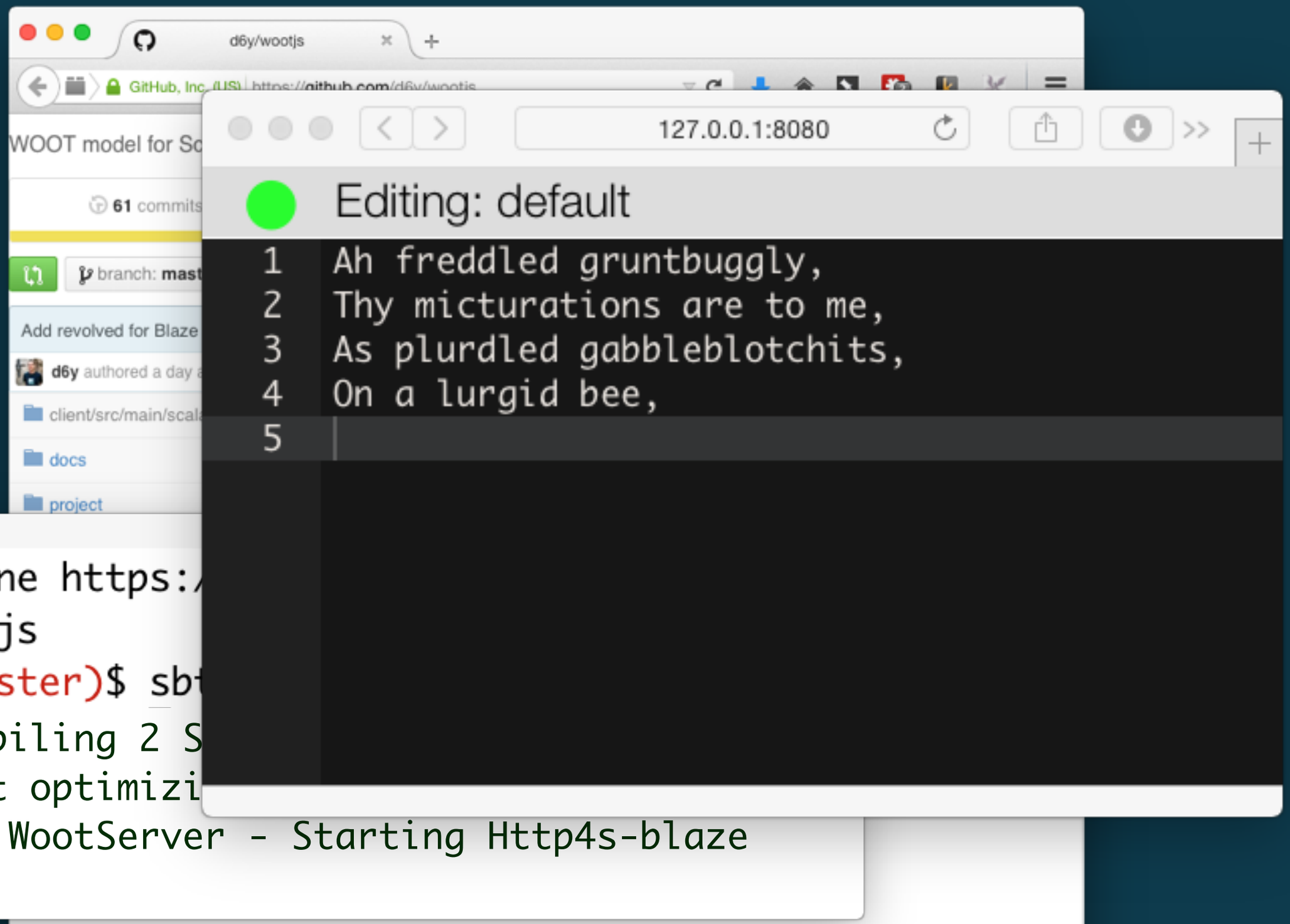
github.com/d6y/wootjs



3. bash

```
~$ git clone https://github.com/d6y/wootjs
~$ cd wootjs
wootjs (master)$ sbt "project server" run
[info] Compiling 2 Scala sources to ...
[info] Fast optimizing woot-client-fastopt.js
2015-05-13 WootServer - Starting Http4s-blaze
```

github.com/d6y/wootjs



Benefits?

IDE support

Single language

Write once, run both places

“It’s the types,
stupid”

Make Change Easier

Two Ideas

Gradually introduce Scala.js
to an existing code base

Distributed data types are
fun & a great fit with Scala.js

Thanks!

Richard Dallaway, @d6y

https://github.com/d6y/wootjs

underscore.io