Scala.js networking made easy

Olivier Blanvillain

PROGRAMMING METHODS LABORATORY, EPFL

January 26, 2015

This Presentation

- 1. Transport library
- 2. Latency compensation framework
- 3. Example: online multiplayer game

Motivation

- Share
- Many JavaScript APIs
- Many network programming models

Diving in...

```
trait Transport {
  type Address
 def listen(): Future[Promise[ConnectionListener]]
 def connect(remote: Address): Future[ConnectionHandle]
 def shutdown(): Future[Unit]
trait ConnectionHandle {
 def handlerPromise: Promise[MessageListener]
 def write(message: String): Unit
 def closedFuture: Future[Unit]
 def close(): Unit
type ConnectionListener = ConnectionHandle => Unit
type MessageListener = String => Unit
```

```
trait Transport {
 type Address
 def listen(): Future[Promise[ConnectionListener]]
 def connect(remote: Address): Future[ConnectionHandle]
 def shutdown(): Future[Unit]
trait ConnectionHandle {
 def handlerPromise: Promise[MessageListener]
 def write(message: String): Unit
 def closedFuture: Future[Unit]
 def close(): Unit
type ConnectionListener = ConnectionHandle => Unit
type MessageListener = String => Unit
```

```
trait Transport {
  type Address
 def listen(): Future[Promise[ConnectionListener]]
 def connect(remote: Address): Future[ConnectionHandle]
 def shutdown(): Future[Unit]
trait ConnectionHandle {
 def handlerPromise: Promise[MessageListener]
 def write(message: String): Unit
 def closedFuture: Future[Unit]
 def close(): Unit
type ConnectionListener = ConnectionHandle => Unit
type MessageListener = String => Unit
```

```
trait Transport {
 type Address
 def listen(): Future[Promise[ConnectionListener]]
 def connect(remote: Address): Future[ConnectionHandle]
 def shutdown(): Future[Unit]
trait ConnectionHandle {
 def handlerPromise: Promise[MessageListener]
 def write(message: String): Unit
 def closedFuture: Future[Unit]
 def close(): Unit
type ConnectionListener = ConnectionHandle => Unit
type MessageListener = String => Unit
```

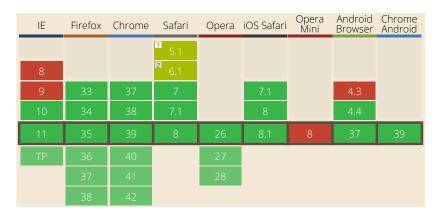
Targeted Technologies

- WebSocket
- SockJS
- WebRTC

WebSocket

Introduction

WebSocket Support



Availability: ~84%

SockJS

Introduction

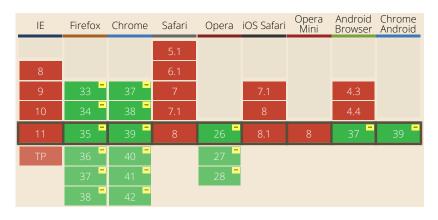
SockJS, Supported Transports

Transport	References		
websocket (rfc6455)	rfc 6455		
websocket (hixie-76)	draft-hixie-thewebsocketprotocol-76		
websocket (hybi-10)	draft-ietf-hybi-thewebsocketprotocol-10		
xhr-streaming	Transport using Cross domain XHR streaming capability (readyState=3).		
xdr-streaming	Transport using XDomainRequest streaming capability (readyState=3).		
eventsource	EventSource.		
iframe-eventsource	EventSource used from an iframe via postMessage.		
htmlfile	HtmlFile.		
iframe-htmlfile	HtmlFile used from an iframe via postMessage.		
xhr-polling	Long-polling using cross domain XHR.		
xdr-polling	Long-polling using XDomainRequest.		
iframe-xhr-polling	Long-polling using normal AJAX from an iframe via postMessage.		
jsonp-polling	Slow and old fashioned JSONP polling.		

WebRTC

Introduction

WebRTC Support



Availability: ~54%

Transport Implementations

Platform	WebSocket	SockJS	WebRTC
JavaScript	client	client	client
Play Framework	server	server	-
Netty	both	-	-
Tyrus	client	-	-

THANKS!