



# Lift-ng: Secure, rapid web development with Scala and AngularJS

Follow along! http://172.30.8.199:8080


Joe Barnes
@joescii
prose :: and :: conz


# Lift-ng: Secure, rapid web development with Scala and AngularJS

Follow along! http://172.30.8.199:8080

**Joe Barnes**

*@joescii*

*prose :: and :: conz*

Primarily Java from 2004-2013
Started Scala late 2012
Discovered lift-ng in Oct 2013
Writing Scala at Mentor Graphics
Lift committer July 2014

What is lift-ng?

Built upon Lift, lift-ng is the most powerful, most secure AngularJS backend available today.

backend available today.

Pretty bold, huh?

Description stolen from Lift:
"Lift is the most powerful, most secure web framework available today."

# Lift is a web framework

Security: Safeguards from 6 OWASP vulnerabilities

Designer-friendly: View-first w/ HTML templates

Outstanding comet and ajax support

Scala's oldest web framework (Feb 2007)

Doesn't hide web development from you

Apache License 2.0

# Angular is a front-end framework

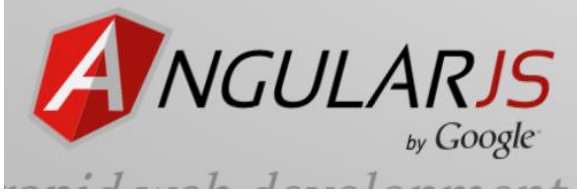
Extends HTML for dynamic web apps

Declarative DOM manipulation

Test-first

Mixes well with Lift

MIT License



## Lift-ng is a Lift module

Provides a Scala DSL for defining:

1. Angular factories
2. `$scope` events and assignments
3. Client/server value bindings (experimental)

Apache License 2.0

# Server Time App

Lift template basics

Angular template basics

JS Angular factory

lift-ng Angular factory

**Time**
Server time at page load:
*Thursday, August 6, 2015 8:49:55 PM CDT*

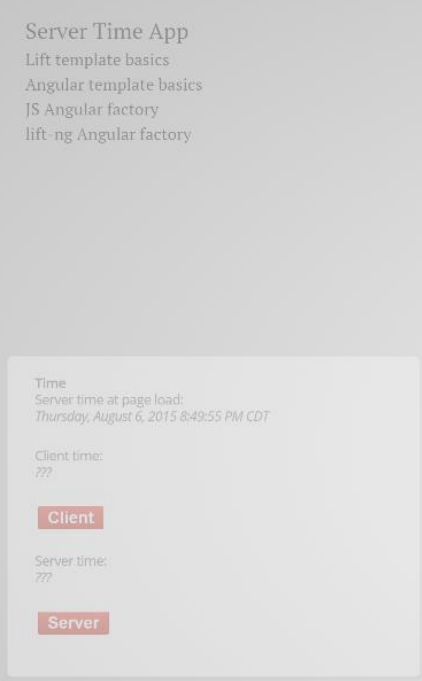
Client time:
*???*

Client

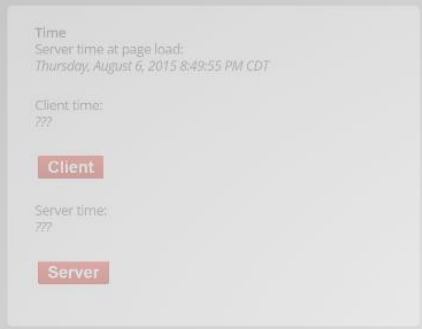
Server time:
*???*

Server

```
<div ng-controller="TimeController">
  <div class="title">Time</div>
  <div>Server time at page load:</div>
  <div data-lift="ServerTime.atPageLoad">
    (span replaced at page load time)
  </div>
  <div>Client time:</div>
  <div class="timestamp" ng-bind="client"></div>
  <button ng-click="getClient()">Client</button>
  <div>Server time:</div>
  <div class="timestamp" ng-bind="server"></div>
  <button ng-click="getServer()">Server</button>
</div>
```



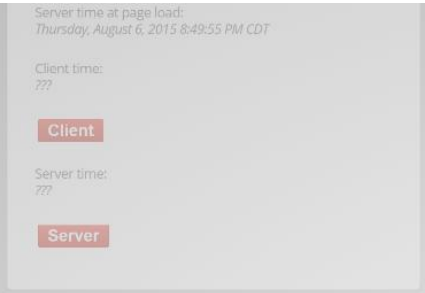

```
private def timestamp =
  dateFormat.format(new Date())

def atPageLoad(template:NodeSeq) =
  <div class="timestamp">
    {timestamp}
  </div>
```




```
angular.module("TimeApp", [
  "ServerTimeServices",
  "ClientTimeServices"
])
```

```
y()
))
```

```
.controller("TimeController", [
  "$scope",
  "ClientTimeService",
  "ServerTimeService",
```




```
    "ServerTimeServices",
    "ClientTimeServices"
  ])


  .controller("TimeController", [
    "$scope",
    "ClientTimeService",
    "ServerTimeService",


  function($scope, clientSvc, serverSvc) {
    // ng-bind="client"
    $scope.client = "???";
```

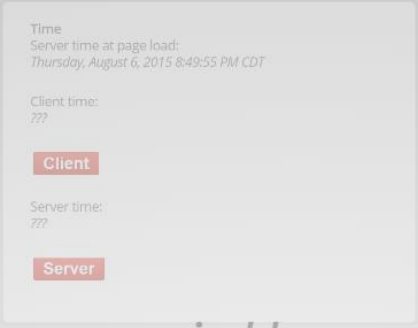
```
ript>
```

```
function($scope, clientSvc, serverSvc) {
  // ng-bind="client"
  $scope.client = "???";
  // ng-bind="server"
  $scope.server = "???";
  // ng-click="getClient()"
  $scope.getClient = function() {
    $scope.client = clientSvc.currentTime()
  };
```

```
angular.module("ClientTimeServices", [])
.factory("ClientTimeService", function() {
  return {
    currentTime: function() {
      return new Date()
    }
  }
})
;
```

```
// ng-click="getServer()"
$scope.getServer = function() {
  serverSvc.currentTime() // promise from server
    .then(function(timestamp) {
      $scope.server = timestamp;
    });
};
```

```
angular.module("ClientTimeServices", [])
 factory("ClientTimeService", function() {
```




```
// lift-ng magic!!
def service = renderIfNotAlreadyDefined(
  angular.module("ServerTimeServices")
  .factory("ServerTimeService", jsObjFactory()
    .jsonCall("currentTime", Full(timestamp))
))
```

```
<script data-lift="ServerTime.service"></script>
```

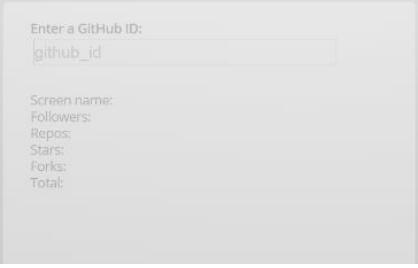
```
def service = renderIfNotAlreadyDefined(
  angular.module("ServerTimeServices")
  .factory("ServerTimeService", jsObjFactory()
    .jsonCall("currentTime", Full(timestamp))
))
```

```
<script data-lift="ServerTime.service"></script>
```

View the source to see the generated angular service

```
<script data-lift="ServerTime.service"></script>
```

View the source to see the generated angular service



# Chat App

Server push via `$scope`

Lift comet via actors

Send us a message

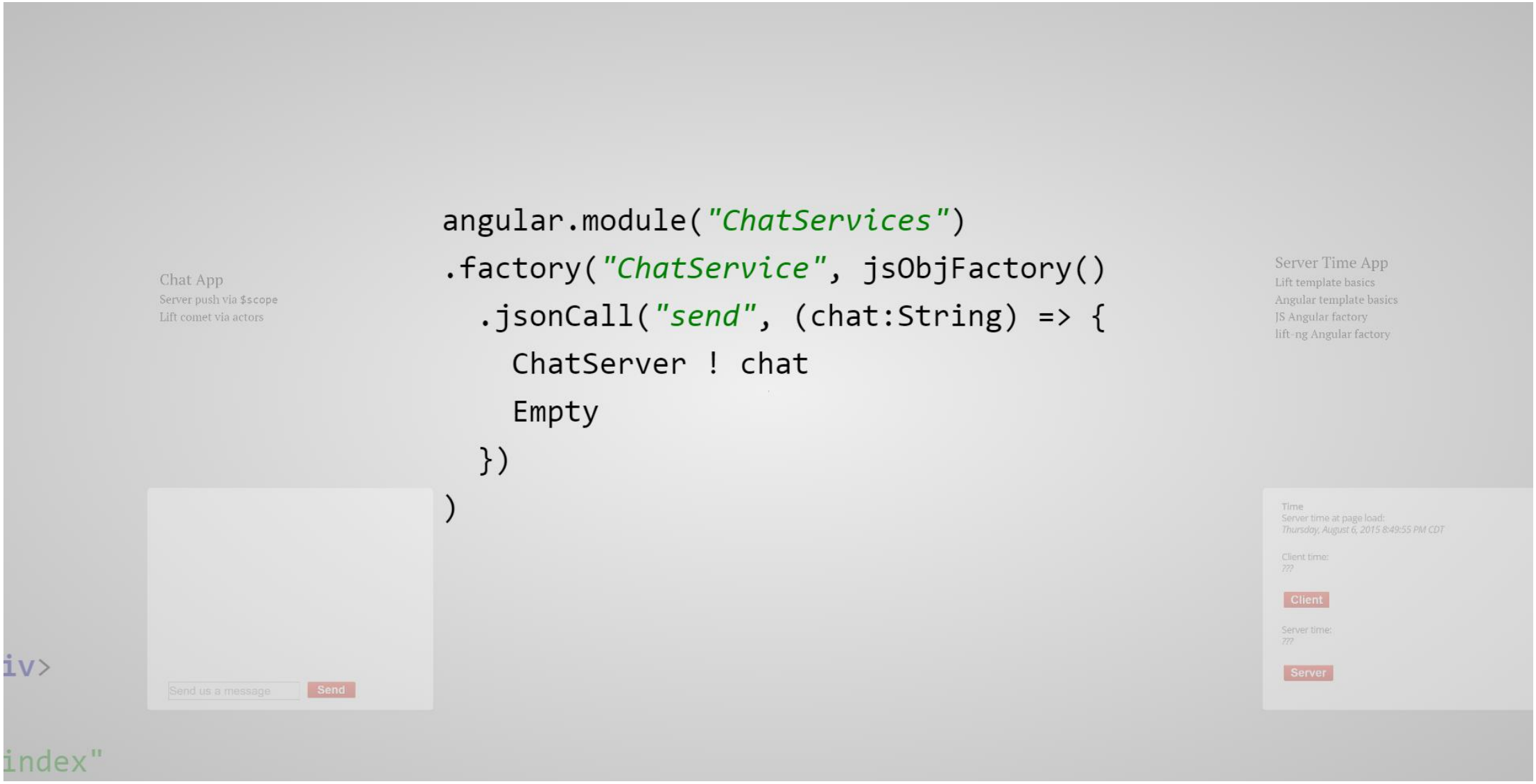
Send

```
$scope.sendChat = function() {
  service.send($scope.message);
  $scope.message = "";
};
```

Chat App

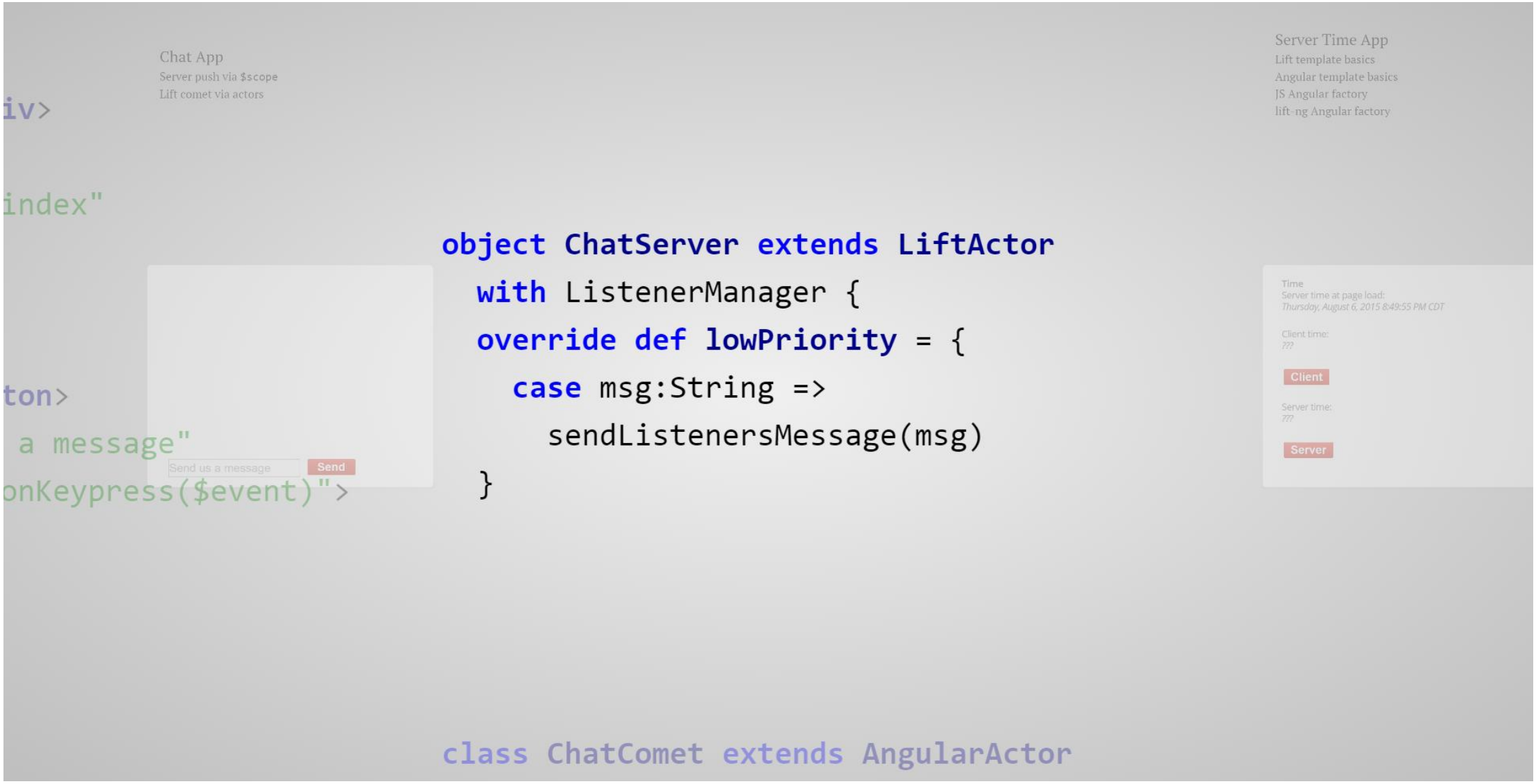
Server push via $scope

Lift comet via actors

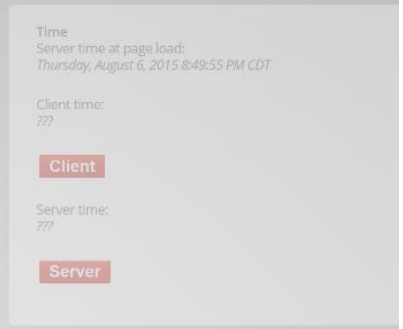



```
angular.module("ChatServices")
.factory("ChatService", jsObjFactory()
  .jsonCall("send", (chat:String) => {
    ChatServer ! chat
    Empty
  })
)
```

```
iv>

index"
```


Chat App
Server push via $scope
Lift comet via actors
Server Time App
Lift template basics
Angular template basics
JS Angular factory
lift-ng Angular factory
iv>
index"
ton>
a message"
onKeypress($event)">
Send us a message
Send
Time
Server time at page load:
Thursday, August 6, 2015 8:49:55 PM CDT
Client time:
???
Client
Server time:
???
Server
object ChatServer extends LiftActor
with ListenerManager {
override def lowPriority = {
case msg:String =>
sendListenersMessage(msg)
}
class ChatComet extends AngularActor

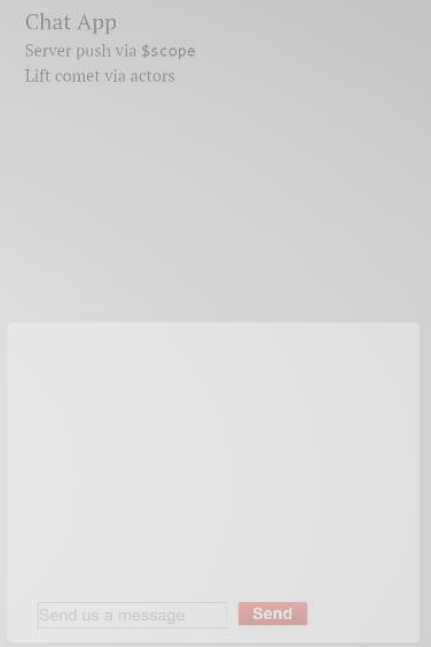

```
class ChatComet extends AngularActor
  with CometListener {
  override def registerWith = ChatServer
  override def lowPriority = {
    case msg:String =>
      scope.emit("new-message", msg)
  }
}
```



```
</div>


$scope.messages = [];
$scope.$on("new-message",
  function(e, msg){
    $scope.messages.push(msg);
  }
);
```

```
<div ng-controller="ChatController">
  <div data-lift="comet?type=ChatComet"></div>
  <ul id="chat-out">
    <li ng-repeat="m in messages track by $index"
        ng-bind="m"></li>
  </ul>
  <div id="chat-in">
    <button ng-click="sendChat()">Send</button>
    <input type="text" placeholder="Send us a message"
           ng-model="message" ng-keypress="onKeypress($event)"
  </div>
</div>
```

View the source to see the comet's DOM



# OSS Score App

Scala Futures -> $q Promises

Enter a GitHub ID:

github_id

Screen name:
Followers:
Repos:
Stars:
Forks:
Total:

```
<div ng-controller="OssScoreController">
  <div class="title">Enter a GitHub ID:</div>
  <input type="text" placeholder="github_id"
         ng-model="enteredId" ng-keypress="onKeypress($event)"

  <div class="avatar" ng-style="{'background-image': 'url('+ava
  <div>Screen name: <span ng-bind="id"></span></div>
  <div>Followers: <span ng-bind="followers"></span></div>
  <div>Repos: <span ng-bind="repos"></span></div>
  <div>Stars: <span ng-bind="stars"></span></div>
  <div>Forks: <span ng-bind="forks"></span></div>
  <div>Total: <span ng-bind="total"></span></div>
</div>
```

OSS Score App

Scala Futures -> $q Promises

View the source to see the generated angular service

# Nothing interesting to see here...

```
// ng-bind="client"
$scope.client = "???";
// ng-bind="server"
$scope.server = "???";
// ng-click="getClient()"
$scope.getClient = function() {
  $scope.client = clientSvc.currentTime()
};
```

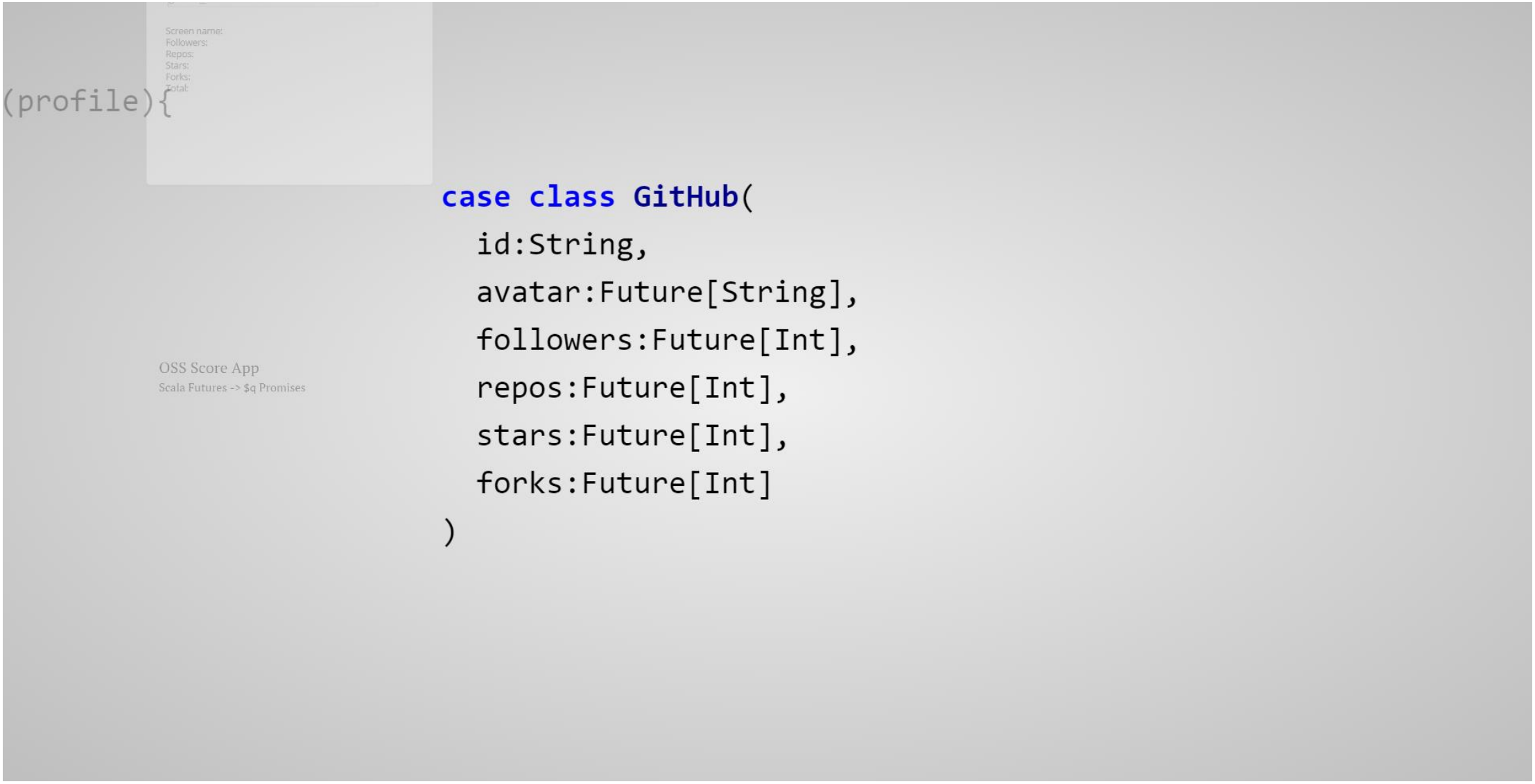
```
<div ng-controller="OssScoreController">
  <div class="title">Enter a GitHub ID:</div>
  <input type="text" placeholder="github_id"
         ng-model="enteredId" ng-keypress="onKeypress($event)">

  <div class="avatar" ng-style="{'background-image': 'url('+avatar+')'}"></div>
  <div>Screen name: <span ng-bind="id"></span></div>
  <div>Followers: <span ng-bind="followers"></span></div>
  <div>Repos: <span ng-bind="repos"></span></div>
  <div>Stars: <span ng-bind="stars"></span></div>
  <div>Forks: <span ng-bind="forks"></span></div>
  <div>Total: <span ng-bind="total"></span></div>
</div>
```

```
service.get($scope.enteredId).then(function(profile){
  $scope.id = profile.id;
  profile.avatar.then(function(avatar){
    $scope.avatar = avatar });
  profile.followers.then(function(count){
    $scope.followers = count });
  profile.repos.then(function(count){
    $scope.repos = count });
  profile.stars.then(function(count){
    $scope.stars = count });
```

```
case class GitHub(
  id:String,
  avatar:Future[String],
  followers:Future[Int],
  repos:Future[Int],
  stars:Future[Int],
  forks:Future[Int]
)
```

```
service.get($scope.enteredId).then(function(profile){
  $scope.id = profile.id;
  profile.avatar.then(function(avatar){
    $scope.avatar = avatar });
  profile.followers.then(function(count){
    $scope.followers = count });
  profile.repos.then(function(count){
    $scope.repos = count });
  profile.stars.then(function(count){
    $scope.stars = count });
  profile.forks.then(function(count){
    $scope.forks = count });
```

```
{
  "id": "joescii",
```


Screen name:
Followers:
Repos:
Stars:
Forks:
Total:
(profile){
OSS Score App
Scala Futures -> $q Promises
case class GitHub(
id:String,
avatar:Future[String],
followers:Future[Int],
repos:Future[Int],
stars:Future[Int],
forks:Future[Int]
)

OSS Score App

Scala Futures -> $q Promises

```
ture":

.future":

ure":

ure":

ure":
```

```
angular.module("GitHubServices")
  .factory("GitHubService", jsObjFactory()
    .jsonCall("get", (github:String) => {
      val gh:GitHub = GitHub.accountFor(github)
      Full(gh)
    })
  )
```

```
{
  "id": "joescii",
  "avatar": {"net.liftmodules.ng.Angular.future":
    "NGU0CXFJYBFXQWUF1RVZ"},
  "followers": {"net.liftmodules.ng.Angular.future":
    "NGPWZRDOWPYUN15QGMGO"},
  "repos": {"net.liftmodules.ng.Angular.future":
    "NGGOYWWGGJ1V1GJXOABW"},
  "stars": {"net.liftmodules.ng.Angular.future":
    "NGZHOHIG0BS4IJ52A5FP"},
  "forks": {"net.liftmodules.ng.Angular.future":
    "NGAERCVUPFXNMP5QGZU3"}
}
```

# Editor App

Binding values between client/server

Experimental!

```
<div ng-controller="EditorController">
  <div data-lift="Angular.bind?type=InputBinder"></div>
  <div data-lift="Angular.bind?type=OutputBinder"></div>
  <div class="document" ng-bind-html="output.dom"></div>
  <textarea ng-model="input.mdtext"></textarea>
</div>
```

Editor App

Binding values between client/server

Experimental!

```
angular.module("EditorApp", ["ngSanitize"])
.controller("EditorController", function(){})
;
```

Editor App

Binding values between client/server

Experimental!

```
case class Input(mdtext:String) extends NgModel
case class Output(dom:String) extends NgModel
```

Editor App

Binding values between client/server

Experimental!

Editor App

Binding values between client/server

Experimental!

```
class InputBinder
  extends SimpleNgModelBinder(
    "input",   // Bind to $scope.input
    Input("")  // Initial value
  ) with BindingToServer {
```

Editor App

Binding values between client/server

Experimental!

```
class OutputBinder
  extends SimpleNgModelBinder(
    "output",  // Bind to $scope.output
    Output(<div></div>)
  ) with BindingToClient with SessionScope
```

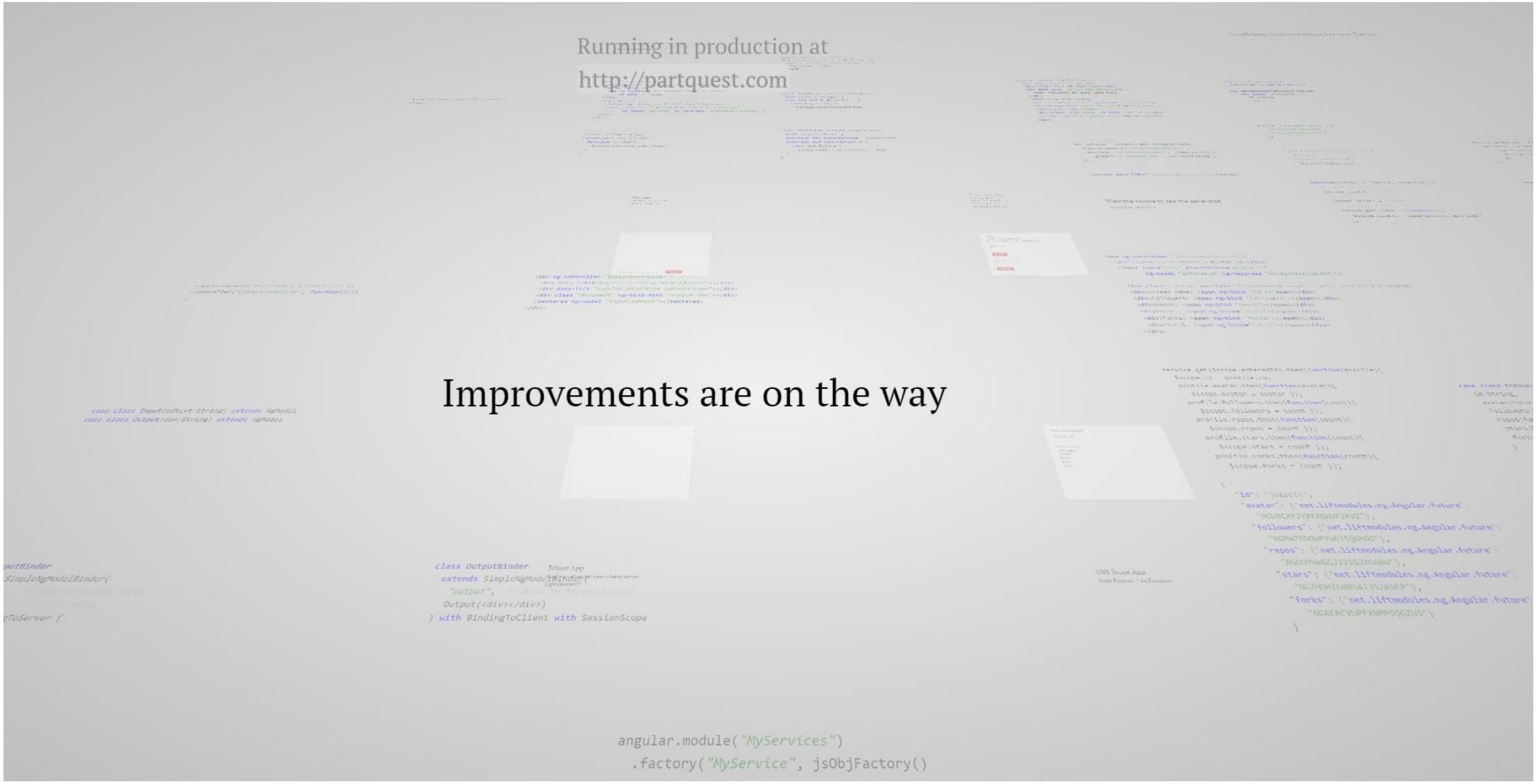
```
override val onClientUpdate =
{ input:Input =>
  for {
    session <- S.session
    content <- MarkdownParser.parse(input.mdtext)
  } {
    session.sendCometActorMessage(
      "OutputBinder",
      Empty, // Comet actors optionally have names
      Output(content)
    )
  }
```


All of this is available now (0.6.4)

All of this is available now (0.6.4)

Running in production at

http://partquest.com


Running in production at
http://partquest.com
Improvements are on the way
class OutputBinder
extends SimpleNgModelBinder(
"output",
Output( )
) with BindingToClient with SessionScope
angular.module("MyServices")
.factory("MyService", jsObjFactory()

```
angular.module("MyServices")
  .factory("MyService", jsObjFactory()
    .defs(callMe = (arg:String) => Service call arg)
    .vals(aConst = "Evaluated at page-load!")
  )
```

Currently can only push to a `$scope`.
Find a way to tie a comet actor to a factory.

Thank you for your interest!

Download Slides

lift-ng

giter8 Template

Presentation Source