



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

Follow along! <http://10.1.4.30:8080>



Joe Barnes

@joescii

prose :: and :: conz



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

Follow along! <http://10.1.4.30:8080>

Joe Barnes

@joescii

prose :: and :: konz

Primarily Java from 2004-2013

Started Scala late 2012

Discovered lift-ng in Oct 2013

Writing Scala at Mentor Graphics

Lift committer July 2014

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."

So what *is* Lift?

MENTION LIFTWEB



**AND EVERYONE LOSES THEIR
MINDS**

memegenerator.net



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



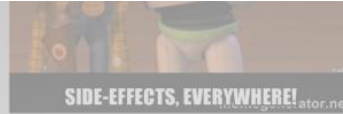
So what is Angular?



SIDE-EFFECTS

SIDE-EFFECTS, EVERYWHERE!

memegenerator.net



Angular is a front-end framework

Extends HTML for dynamic web apps

Declarative DOM manipulation

Test-first

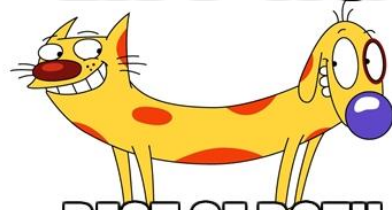
Mixes well with Lift

MIT License



So then what is lift-ng?

LIFT-NG



**BEST OF BOTH
WORLDS?**

memegenerator.net

BEST OF BOTH
WORLDS?

memegenerator.net

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:

Saturday, August 15, 2015 11:45:06 AM 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">
    (this div 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>

```

Server Time App

Lift template basics

Angular template basics

JS Angular factory

lift-ng Angular factory

Time
Server time at page load:
Saturday, August 15, 2015 11:45:06 AM CDT

Client time:
???

Client

Server time:
???

Server

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

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

Time
Server time at page load:
Saturday, August 15, 2015 11:45:06 AM CDT

Client time:
???

Client

Server time:
???

Server

```
angular.module("TimeApp", [  
    "ClientTimeModule",  
    "ServerTimeModule"  
])
```

```
.controller("TimeController", [  
    "$scope",  
    "ClientTime",  
    "ServerTime",  
])
```

Server time at page load:
Saturday, August 15, 2015 11:45:06 AM CDT

Client time:
???

Client

Server time:
???

Server

```
"ClientTimeModule",  
"ServerTimeModule"  
])
```

```
.controller("TimeController", [  
    "$scope",  
    "ClientTime",  
    "ServerTime",
```

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

ript>

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

Enter a GitHub ID:

github_id

Screen name:

Followers:

Repos:

Stars:

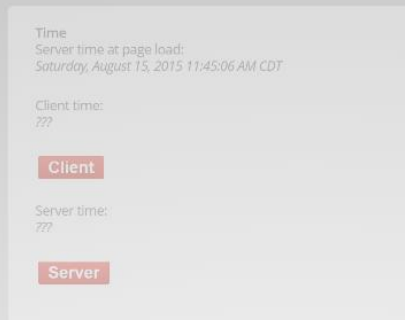
Forks:

Total:

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

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

```
angular.module("ClientTimeModule", [])
    .factory("ClientTime", function() {
```



// lift-ng magic!!

```
def service = renderIfNotAlreadyDefined(  
  angular.module("ServerTimeModule")  
    .factory("ServerTime", jsonObjFactory()  
      .jsonCall("currentTime", Full(timestamp))  
    )  
)
```

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



```
def service = renderIfNotAlreadyDefined(  
  angular.module("ServerTimeModule")  
    .factory("ServerTime", jsonObjFactory()  
      .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

Enter a GitHub ID:

github_id

Screen name:

Followers:

Repos:

Stars:

Forks:

Total:

Chat App

Server push via `$scope`

Lift comet via actors

Send us a message

Send

Chat App

Server must be in scope
of the controller

```
<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>
```

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

Chat App
Server push via \$scope
Lift comet via actors

Chat App
Server push via \$scope
Lift comet via actors

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

iv>

index"

Send us a message

Server Time App
Lift template basics
Angular template basics
JS Angular factory
lift-ng Angular factory

Time
Server time at page load:
Saturday, August 15, 2015 11:45:06 AM CDT

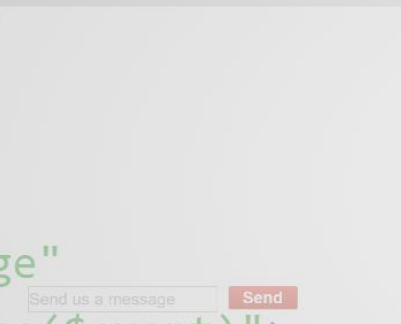
Client time:
???

Server time:
???

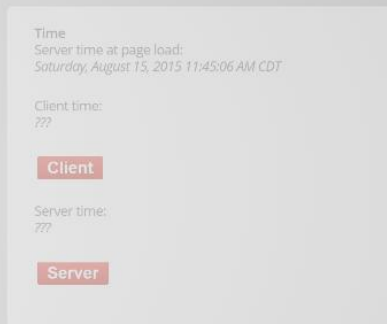
Chat App
Server push via \$scope
Lift comet via actors

Server Time App
Lift template basics
Angular template basics
JS Angular factory
lift-ng Angular factory

```
object ChatServer extends LiftActor
with ListenerManager {
  override def lowPriority = {
    case msg:String =>
      sendListenersMessage(msg)
  }
}
```



A screenshot of a web form. It features a light gray rectangular box containing a text input field with the placeholder text "Send us a message" and a red "Send" button to its right.



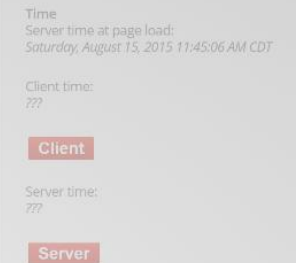
A screenshot of a web page titled "Time". It displays "Server time at page load: Saturday, August 15, 2015 11:45:06 AM CDT". Below this, it shows "Client time: ???". There are two red buttons: "Client" and "Server". At the bottom, it shows "Server time: ???".

```
class ChatComet extends AngularActor
```




Send us a message

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

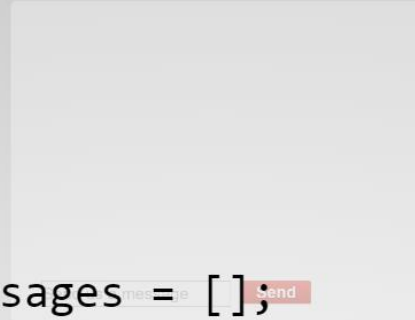


Time
Server time at page load:
Saturday, August 15, 2015 11:45:06 AM CDT

Client time:
???

Server time:
???

```
</div>
```



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

View the source to see the comet's
DOM

Chat App
Server push via \$scope
Lift comet via actors

Send us a message Send

OSS Score App

Scala Futures -> \$q Promises

Enter a 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>

```

View the source to see the generated
angular service

```
// ng-init= client
$scope.client = "???";
// ng-bind=server
$scope.server = "???";
// ng-click=getClient()
$scope.getClient = function() {
    $scope.client = clientTime.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>
```

Nothing interesting to see here...

```
OSS Score App
Scala Futures > fa Promises
github.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]
)
```

```
github.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",
```



```
profile){
```

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

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":

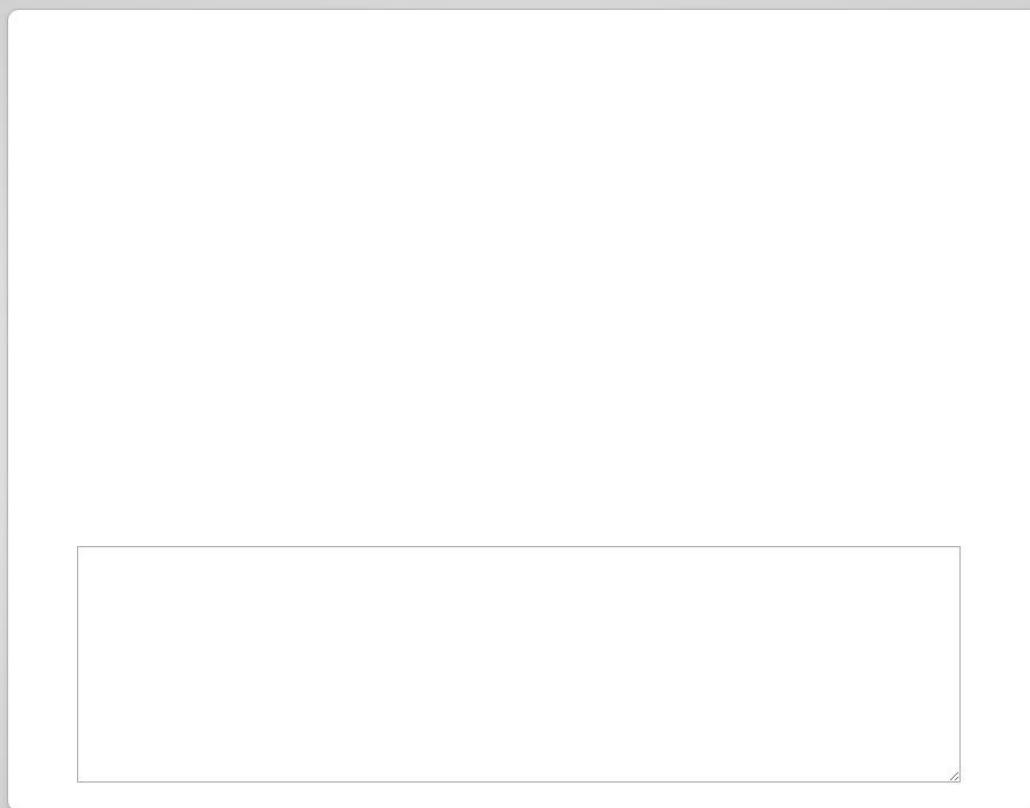
```
angular.module("GitHubModule")
  .factory("GitHub", jsonObjFactory()
    .jsonCall("get", (github:String) => {
      val gh:GitHub = GitHub.accountFor(github)
      Full(gh)
    })
  )
```

```
{
  "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"])\n.controller("EditorController", function(){})\n;
```

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!


```
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)  
    )  
  }  
}
```

Support for i18n

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

```
object ChatServer extends LiftActor
  with ListenerManager {
    override def lowPriority = {
      case msg:String =>
        sendListenersMessage(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 ng-controller="TimeController">
  <div class="title">Time</div>
  <div>Server time at page load:</div>
  <div data-lift="ServerTime.atPageLoad">
    (this div replaced at page load th
  </div>
  <div>Client time:</div>
  <div class="timestamp" ng-bind-
  <button ng-click="getClient()">
  <div>Server time:</div>
  <div class="timestamp" ng
```

Support for i18n

```
# my-bundle.properties
```

```
hello=¡Hola!
```

```
bye=Adios, {0}
```

```
<div ng-controller="chatController">
  <div data-lift="{ chatType:'chatForm' }"></div>
  <ui:el id="chat-out">
    {{ ng-repeat="m in messages track by $index"
      ng-bind="m" }}</li>
  </ui>
  <div id="chat-in">
    <button ng-click="sendChat()">Send</button>
    <input type="text" placeholder="Send us a message"
      ng-model="message" ng-keypress="onKeyPress($event)">
  </div>
</div>
```

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

```
object ChatServer extends LiftActor
  with ListenerManager {
    override def lowPriority = {
      case msg:String =>
        sendListenersMessage(msg)
    }
  }
```

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

```
// life-ng module
def service = renderIfNotAlready
angular.module("ServerTimeMod", [])
.factory("ServerTime", jsObj
.jsonCall("currentTime"
))
```

```
<script data-lift=">
```

```
# my-bundle.properties
hello=¡Hola!
bye=Adios, {0}
```

```
angular.module('ExampleApp', ['i18n'])
.controller('ExampleController',
  ['$scope', 'my-bundle',
    function($scope, i18n) {
      $scope.hello = i18n.hello;
      $scope.bye = i18n.bye($scope.username);
    }
  ]
);
```

View the source to see the exact's DOM

```
$scope.message = null;
$scope.$on('new-message',
  function(e, msg){
    $scope.message = msg;
  }
);
```

Send me a message

Send

Send me a message

Send me a message

Send

```
def service = renderWithViewPartials(
  angular.module('ServerTimeModule')
    .factory('ServerTime', $provide.factory())
    .install('currentTime', null(timestamp))
);
```

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

View the source to see the generated angular service

```
<div ng-controller="OssScores
  <div class="title">Enter
```


All of this is available now (0.7.0)

```

<div ng-controller="testController">
  <div class="form">
    <input type="text" placeholder="placeholder" ng-model="entered" ng-keypress="onKeyPress($event)" />
  </div>
</div>

```

[illegible]

1. **Introduction**
 2. **Background**
 3. **Methodology**
 4. **Results**
 5. **Conclusion**
 6. **References**
 7. **Appendix**
 8. **Index**
 9. **Glossary**
 10. **Notes**
 11. **References**
 12. **Appendix**
 13. **Index**
 14. **Glossary**
 15. **Notes**
 16. **References**
 17. **Appendix**
 18. **Index**
 19. **Glossary**
 20. **Notes**
 21. **References**
 22. **Appendix**
 23. **Index**
 24. **Glossary**
 25. **Notes**
 26. **References**
 27. **Appendix**
 28. **Index**
 29. **Glossary**
 30. **Notes**
 31. **References**
 32. **Appendix**
 33. **Index**
 34. **Glossary**
 35. **Notes**
 36. **References**
 37. **Appendix**
 38. **Index**
 39. **Glossary**
 40. **Notes**
 41. **References**
 42. **Appendix**
 43. **Index**
 44. **Glossary**
 45. **Notes**
 46. **References**
 47. **Appendix**
 48. **Index**
 49. **Glossary**
 50. **Notes**
 51. **References**
 52. **Appendix**
 53. **Index**
 54. **Glossary**
 55. **Notes**
 56. **References**
 57. **Appendix**
 58. **Index**
 59. **Glossary**
 60. **Notes**
 61. **References**
 62. **Appendix**
 63. **Index**
 64. **Glossary**
 65. **Notes**
 66. **References**
 67. **Appendix**
 68. **Index**
 69. **Glossary**
 70. **Notes**
 71. **References**
 72. **Appendix**
 73. **Index**
 74. **Glossary**
 75. **Notes**
 76. **References**
 77. **Appendix**
 78. **Index**
 79. **Glossary**
 80. **Notes**
 81. **References**
 82. **Appendix**
 83. **Index**
 84. **Glossary**
 85. **Notes**
 86. **References**
 87. **Appendix**
 88. **Index**
 89. **Glossary**
 90. **Notes**
 91. **References**
 92. **Appendix**
 93. **Index**
 94. **Glossary**
 95. **Notes**
 96. **References**
 97. **Appendix**
 98. **Index**
 99. **Glossary**
 100. **Notes**
 101. **References**
 102. **Appendix**
 103. **Index**
 104. **Glossary**
 105. **Notes**
 106. **References**
 107. **Appendix**
 108. **Index**
 109. **Glossary**
 110. **Notes**
 111. **References**
 112. **Appendix**
 113. **Index**
 114. **Glossary**
 115. **Notes**
 116. **References**
 117. **Appendix**
 118. **Index**
 119. **Glossary**
 120. **Notes**
 121. **References**
 122. **Appendix**
 123. **Index**
 124. **Glossary**
 125. **Notes**
 126. **References**
 127. **Appendix**
 128. **Index**
 129. **Glossary**
 130. **Notes**
 131. **References**
 132. **Appendix**
 133. **Index**
 134. **Glossary**
 135. **Notes**
 136. **References**
 137. **Appendix**
 138. **Index**
 139. **Glossary**
 140. **Notes**
 141. **References**
 142. **Appendix**
 143. **Index**
 144. **Glossary**
 145. **Notes**
 146. **References**
 147. **Appendix**
 148. **Index**
 149. **Glossary**
 150. **Notes**
 151. **References**
 152. **Appendix**
 153. **Index**
 154. **Glossary**
 155. **Notes**
 156. **References**
 157. **Appendix**
 158. **Index**
 159. **Glossary**
 160. **Notes**
 161. **References**
 162. **Appendix**
 163. **Index**
 164. **Glossary**
 165. **Notes**
 166. **References**
 167. **Appendix**
 168. **Index**
 169. **Glossary**
 170. **Notes**
 171. **References**
 172. **Appendix**
 173. **Index**
 174. **Glossary**
 175. **Notes**
 176. **References**
 177. **Appendix**
 178. **Index**
 179. **Glossary**
 180. **Notes**
 181. **References**
 182. **Appendix**
 183. **Index**
 184. **Glossary**
 185. **Notes**
 186. **References**
 187. **Appendix**
 188. **Index**
 189. **Glossary**
 190. **Notes**
 191. **References**
 192. **Appendix**
 193. **Index**
 194. **Glossary**
 195. **Notes**
 196. **References**
 197. **Appendix**
 198. **Index**
 199. **Glossary**
 200. **Notes**
 201. **References**
 202. **Appendix**
 203. **Index**
 204. **Glossary**
 205. **Notes**
 206. **References**
 207. **Appendix**
 208. **Index**
 209. **Glossary**
 210. **Notes**
 211. **References**
 212. **Appendix**
 213. **Index**
 214. **Glossary**
 215. **Notes**
 216. **References**
 217. **Appendix**
 218. **Index**
 219. **Glossary**
 220. **Notes**
 221. **References**
 222. **Appendix**
 223. **Index**
 224. **Glossary**
 225. **Notes**
 226. **References**
 227. **Appendix**
 228. **Index**
 229. **Glossary**
 230. **Notes**
 231. **References**
 232. **Appendix**
 233. **Index**
 234. **Glossary**
 235. **Notes**
 236. **References**
 237. **Appendix**
 238. **Index**
 239. **Glossary**
 240. **Notes**
 241. **References**
 242. **Appendix**
 243. **Index**
 244. **Glossary**
 245. **Notes**
 246. **References**
 247. **Appendix**
 248. **Index**
 249. **Glossary**
 250. **Notes**
 251. **References**
 252. **Appendix**
 253. **Index**
 254. **Glossary**
 255. **Notes**
 256. **References**
 25

```
<div ng-controller="EditorController">  
  <div data-lift="ngula" kind="input"></div>  
  <div data-lift="angular" type="text"></div>  
</div>
```

Running in production at
<http://partquest.com>

```
Input(motext:String) extends NgModel
input(dom:String) extends NgModel
```


Running in production at
<http://partquest.com>

Improvements are on the way

```
angular.module("MyModule")  
  .factory("MyFactory", jsonObjFactory())
```

```
angular.module("MyModule")  
  .factory("MyFactory", jsonObjFactory()  
    .defs(callServer = (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.

And maybe we'll dig into Angular 2

Every feature is covered with either a unit test or Selenium integration test. I'm keenly interested in not breaking anything.

Thank you for your interest!

[Download Slides](#)

[lift-ng](#)

[giter8 Template](#)

[Presentation Source](#)