Hybrid View Rendering with Grails

Rob Fletcher

@rfletcherEW

NETFLIX

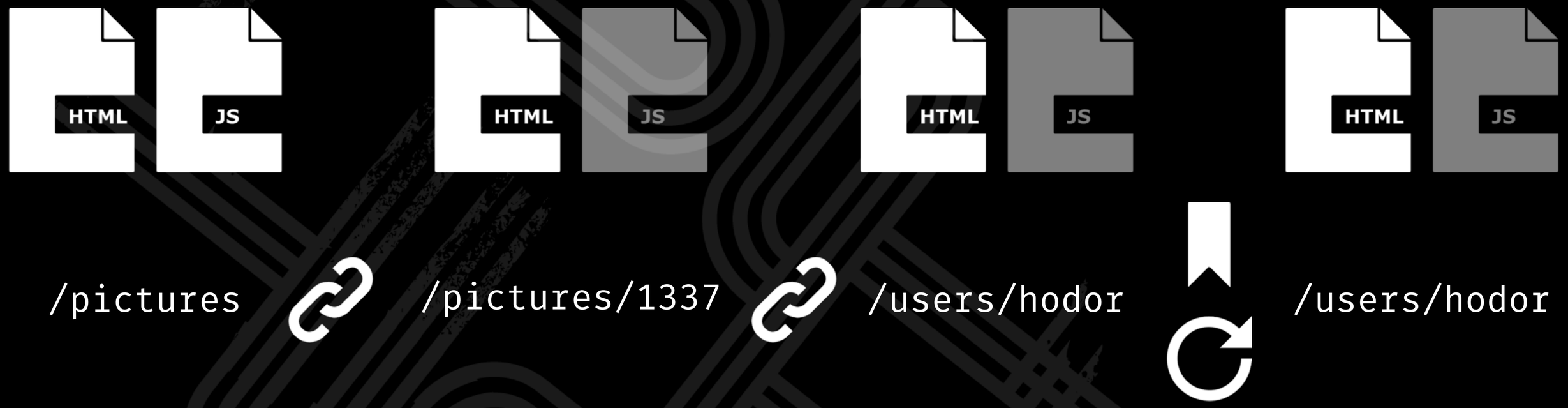
GR Conf EUROPE 2014

Server, Client & Hybrid

- rendered on the server
- serves HTML document
- e.g. static files, PHP, GSP

- rendered in the browser
- JSON over REST
- e.g. Backbone, Ember

- rendered on the server **or** in the browser
- serves HTML or JSON



“Traditional” Server-Side Views

e.g. static files, *cgi-bin* or GSP

One Page Apps

e.g. Backbone, Ember

Hybrid Views

Combine the advantages

Why bother?

This sounds like a lot of work

Twitter...

"spent *a year and 40 engineers* rebuilding their site to render on the server instead of the client, claiming a *5x improvement* in perceived loading time"

– Isomorphic JavaScript

Amazon...

"increased revenue by *1%* for *every 100 milliseconds* of improvement"

– Strangeloop

100
100
100
100
100
100
100
100
LEGO

History

Apr 2012: Show HN: Meteor [link]

May 2012: Twitter announces rebuild with server-side rendering [link]

Jul 2012: Meteor receives $11.2m VC funding [link]

Dec 2012: Grails for Hipsters talk at GGX [link]

Nov 2013: Isomorphic JavaScript: The Future of Web Apps [link]

Node.js Frameworks

Derby

Mojito

Rendr

Template Options

{{ it's all about the facial hair }}

Mustache

Handlebars

Hogan

```
1 <h1>Timeline</h1>
2 <ul>
3   {{#each .}}
4   <li data-id="{{id}}">
5     {{> picture}}
6   </li>
7   {{/each}}
8 </ul>
```

```
1 <figure>
2 <img src="{{url}}">
3 <figcaption>
4   <p>Uploaded {{friendlyTime dateCreated}} by {{uploadedBy.username}}</p>
5   {{#isCurrentUser uploadedBy}}
6   <button type="button" class="delete" title="Delete">Delete</button>
7   {{/isCurrentUser}}
8 </figcaption>
9 </figure>
```

#!☠†ing URLs

http://hipsteroid.io/pictures/1

http://hipsteroid.io/#!/pictures/1

http://hipsteroid.io/?_escaped_fragment_=/pictures/1

History.pushState

May 2010

November 2010

March 2011

June 2011

September 2012

October 2012

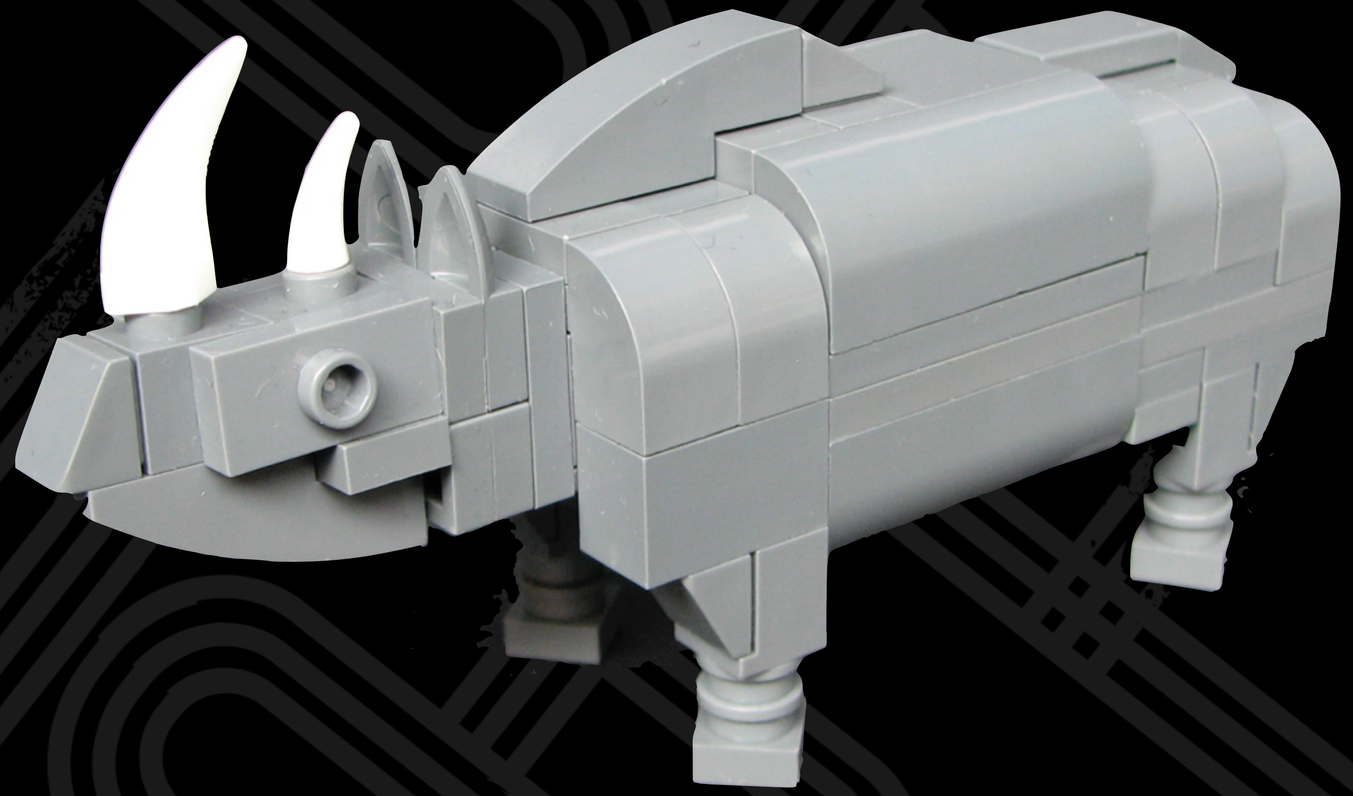
Implementation with Grails

Handlebars Plugin

http://git.io/grails-handlebars

Handlebars Asset Pipeline

http://git.io/handlebars-asset-pipeline



Nashorn

High-performance JavaScript engine in the JVM

```
1  AssetResourceLocator assetResourceLocator
2
3  @PostConstruct
4  void initialize() {
5    def engineManager = new ScriptEngineManager()
6
7    def engine = engineManager.getEngineByName("javascript")
8
9    def resource = assetResourceLocator.findResourceForURI("handlebars.js")
10   resource.inputStream.withReader { Reader reader ->
11     engine.eval(reader)
12   }
13 }
```

Injected by asset-pipeline plugin

Load the JavaScript engine:
Java 8: Nashorn
Java 6-7: Rhino

use asset-pipeline to find the Handlebars JavaScript library

Parse the Handlebars JavaScript library

```
1  def render = { attrs ->
2    if (!attrs.template) throwTagError("'template' attribute is required")
3    if (!attrs.model) throwTagError("'model' attribute is r
4
5    def path = "${templatesRoot}/${attrs.template}.js"
6    def resource = assetResourceLocator.findResourceForURI(path)
7    resource.inputStream.withReader { Reader reader ->
8      engine.eval(reader)
9    }
10
11   def templates = engine.eval("Handlebars.templates")
12   out << engine.invokeMethod(templates, attrs.template, attrs.model)
13 }
```

find and parse the requested Handlebars template

Get the Handlebars templates – our template is a function on that object

target object

function name

parameters

```
1  def list() {
2    def photos = Photo.list(params)
3
4    withFormat {
5      json {
6        render photos as JSON
7      }
8      html {
9        render contentType: "text/html",
10              text: hbs.render(template: "photos", model: [photos: photos])
11     }
12   }
13 }
```

Render the model as JSON

Render HTML using a Handlebars template

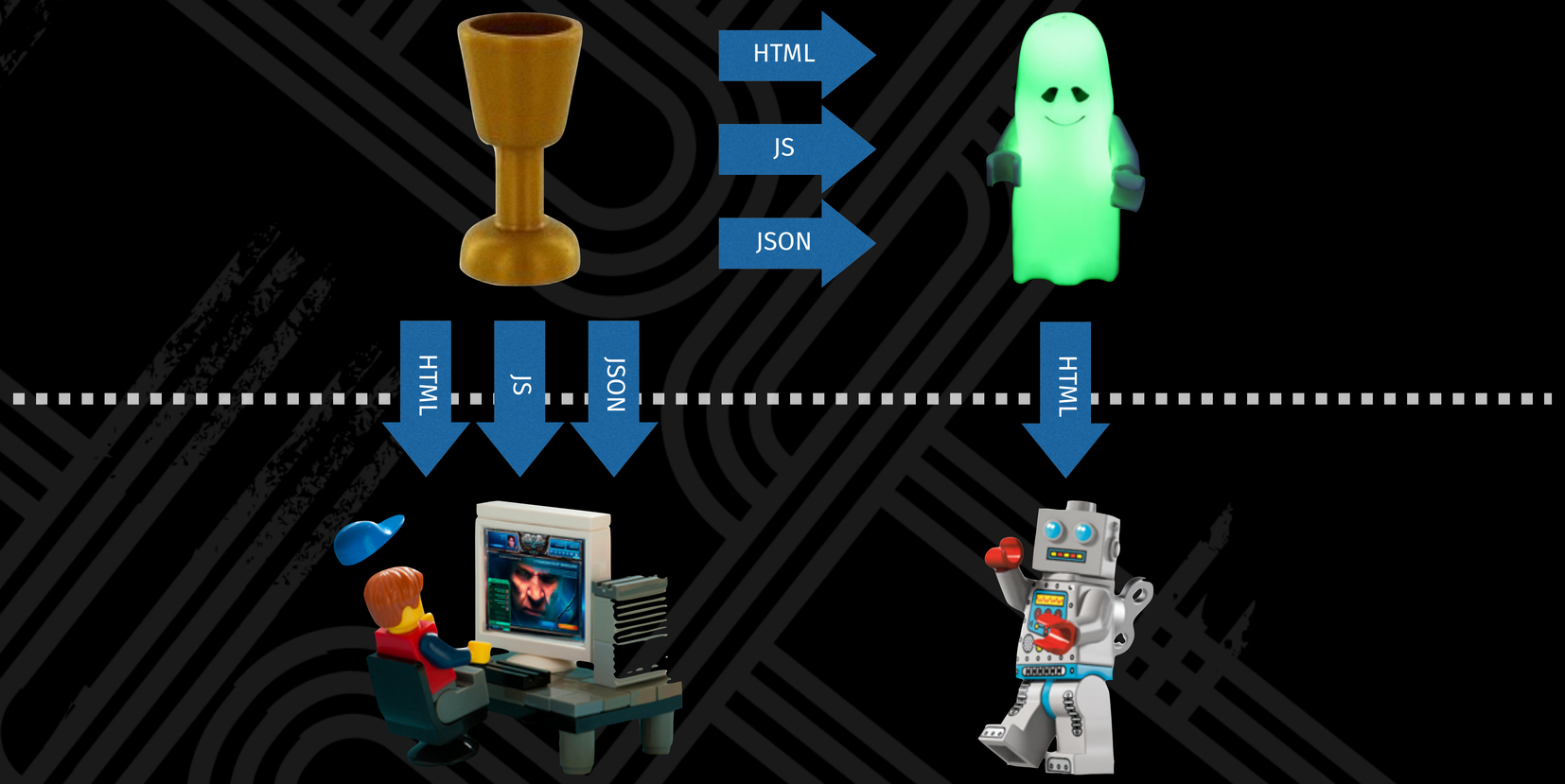
Mirrored routing

```
"/" {
  controller = "home"
}

"/pictures" {
  controller = "pictures"
  action = "list"
}

"/pictures/$id" {
  controller = "pictures"
  action = "show"
}

"/users/$name" {
  controller = "user"
  action = "show"
}
```

```
routes: {
  '/'              : 'home',
  '/pictures'      : 'pictures',
  '/pictures:id'   : 'picture',
  '/users:name'    : 'profile'
}
```

We Can Build It!

But, what about...?



Headless Browser Proxy

Phantom JS on the server

Embed JSON in initial page

```
<script>
  var initialPayload = [
    {"image": "/pictures/1.jpg", "uploadedBy": "/users/hodor"},
    {"image": "/pictures/2.jpg", "uploadedBy": "/users/theon"},
    {"image": "/pictures/3.jpg", "uploadedBy": "/users/joffrey"}
  ]
</script>
```

Load JSON Synchronously

```
<script src="/pictures.jsonp"></script>
```

returns…

```
loadData([
  {"image": "/pictures/1.jpg", "uploadedBy": "/users/hodor"},
  {"image": "/pictures/2.jpg", "uploadedBy": "/users/theon"},
  {"image": "/pictures/3.jpg", "uploadedBy": "/users/joffrey"}
]);
```

Yes, I you a question…