28. November 2008

Johannes Link

Coach für agile Softwareentwicklung


Johannes Link
Unter Mitarbeit von Frank Adler, Achim Bangert, Ekard Burger, Peter Fröhlich, Ilja Preuß
Softwaretests mit JUnit
Techniken der testgetriebenen Entwicklung
→ 2., überarbeitete und erweiterte Auflage
dpunkt.verlag

Heidelberg


Hamburg
Szczecin
Bremen
Berlin
Hannover
Bielefeld
Essen
Leipzig
Deutschland
Dresden
Köln
Frankfurt
am Main
Mannheim
Nürmberg
Karlsruhe
Stuttgart
Strasbourg
München
Österreich

GermanHome **Home** Offer ClasspathSuite MockMe ReFit

Johannes Link

Coach for Agile Development. Software Activist. Programmer.

English Stuff

- I just released **MockMe for JavaScript** - a new mocking library for JavaScript. The documentation is not fully done yet, but I wanted it published before I leave for **Toronto**.
- **ReFit** - a tool for refactoring FitNesse test pages is online. Read **how to use it** or **my blog post on the topic**.
- Version 1.1.0 of **ClasspathSuite for JUnit 4** has been released. It now works with JUnit 4.4 and comes with support for old style tests aka JUnit 3.8 test cases.
- Meanwhile most of my material (for workshops and tutorials) is **available in English**.

MyGermanBook MyEnglishBook  Weblog EMail

GermanHome **Home** Offer ClasspathSuite MockMe ReFit

johanneslink.net

English Stuff

- I just released **MockMe for JavaScript** - a new mocking library for JavaScript. The documentation is not fully done yet, but I wanted it published before I leave for **Toronto**.
- **ReFit** - a tool for refactoring FitNesse test pages is online. Read **how to use it** or **my blog post on the topic**.
- Version 1.1.0 of **ClasspathSuite for JUnit 4** has been released. It now works with JUnit 4.4 and comes with support for old style tests aka JUnit 3.8 test cases.
- Meanwhile most of my material (for workshops and tutorials) is **available in English**.

MyGermanBook MyEnglishBook Weblog EMail

Ajax und Web 2.0: Die Grenze der testgetriebenen Entwicklung?

Asynchronous
JAvaScript and
XML


IE6

Security
Firefox
Chrome
IE
jQuery
JsMock
Gears
DWR
Air
Opera
ActionScript
Video
Mobile
YUI
Flex
XML
JSON
JSF
Sound
J3Unit
GoogleMaps
iPhone
WebTest
prototype
JSUnit
WebKit
RAP
SVG
XBL
MooTools
Jayjax
JSSpec
GWT
ColdFusion
HeatMapAPI
JavaScript
Accessibility
Greasemonkey
Flash
XHR
Gecko
Rails
Silverlight
script.aculo.us
Safari
MashUp
Java
jackson
JavaFX
MockMe
Ext
IE PNG Fix
Grails
JsTester
Comet
Microformat
dojo
json-lib
Selenium
Firebug
JavaScriptMVC

Security
Firefox
Chrome
IE
jQuery
JsMock
Gears
DWR
Air
Opera
ActionScript
Video
Mobile
YUI
Flex
XML
JSON
JSF
Sound
J3Unit
GoogleMaps
iPhone
WebTest
prototype
JSUnit
WebKit
RAP
SVG
XBL
MooTools
Jayjax
JSSpec
GWT
ColdFusion
HeatMapAPI
JavaScript
Accessibility
Greasemonkey
Flash
XHR
Gecko
Rails
Silverlight
script.aculo.us
Safari
MashUp
Java
jackson
JavaFX
MockMe
Ext
IE PNG Fix
Grails
JsTester
Comet
Microformat
dojo
json-lib
Selenium
Firebug
JavaScriptMVC

Firefox Chrome

JSON

GoogleMaps

prototype

JavaScript

XHR

script.aculo.us Java

MockMe

jackson

Selenium

Firefox

JSON

GoogleMaps

prototype

JavaScript

XHR

script.aculo.us

Java

MockMe

jackson

Selenium

Die teilweise testgetriebene Entwicklung einer einfachen Web-Applikation, die JavaScript, XHR, DOM-Manipulation, prototype und script.aculo.us verwendet und mit GoogleMaps (tm) „gemasht“ wird. Und all das aus der Perspektive eines Java-Entwicklers.

Off-Topic

Off-Topic

- What is Test-Driven Development

Off-Topic

- What is Test-Driven Development
- TDD vs BDD

Off-Topic

- What is Test-Driven Development
- TDD vs BDD
- 100+ Tools und Frameworks

Agenda

Agenda

- Herausforderungen

Agenda

- Herausforderungen
- Techniken & Werkzeuge

Agenda

- Herausforderungen
- Techniken & Werkzeuge
  - JavaScript Unit Testing

Agenda

- Herausforderungen
- Techniken & Werkzeuge
  - JavaScript Unit Testing
  - Akzeptanztests

Agenda

- Herausforderungen
- Techniken & Werkzeuge
  - JavaScript Unit Testing
  - Akzeptanztests
- ELO: Event Location Optimizer

Agenda

- Herausforderungen
- Techniken & Werkzeuge
  - JavaScript Unit Testing
  - Akzeptanztests
- ELO: Event Location Optimizer
- Lessons Learned

Server

Tomcat, Rails...

Java, PHP...

Servlets
JSF
DWR (server lib)

Web Client

HTML + CSS

JavaScript

prototype
script.aculo.us
Dojo
DWR (client lib)

Server

Tomcat, Rails...

Java, PHP...

Servlets
JSF
DWR (server lib)

Web Client

HTML + CSS

JavaScript

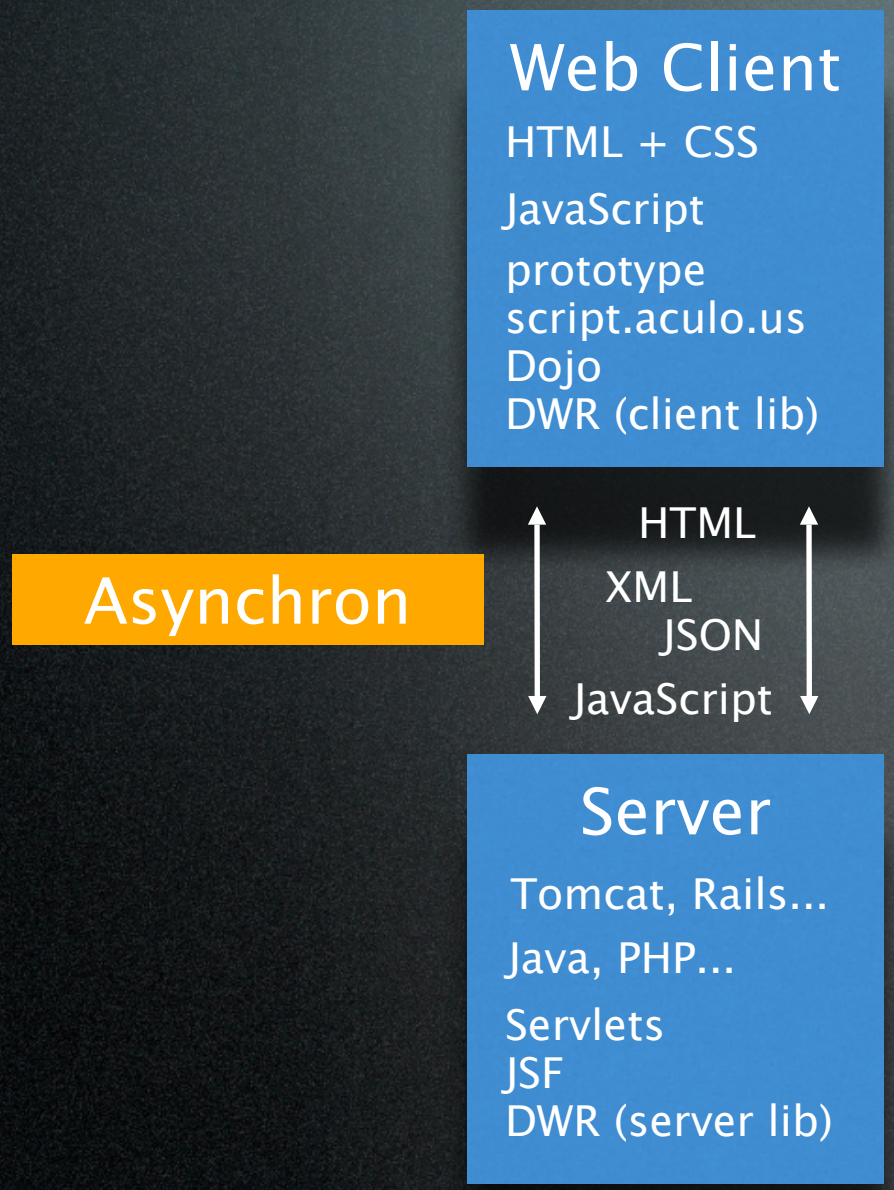
prototype
script.aculo.us
Dojo
DWR (client lib)

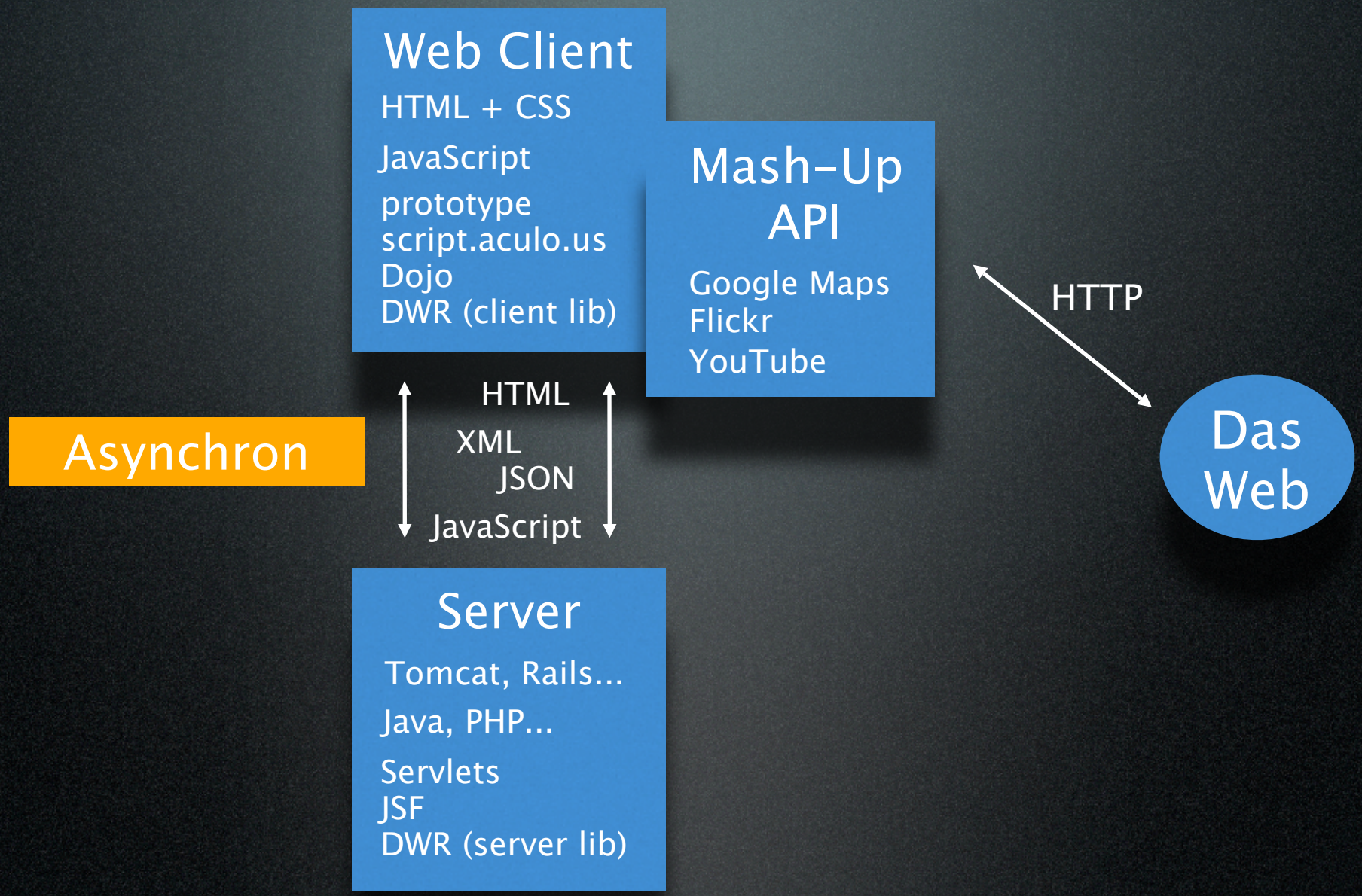
Server

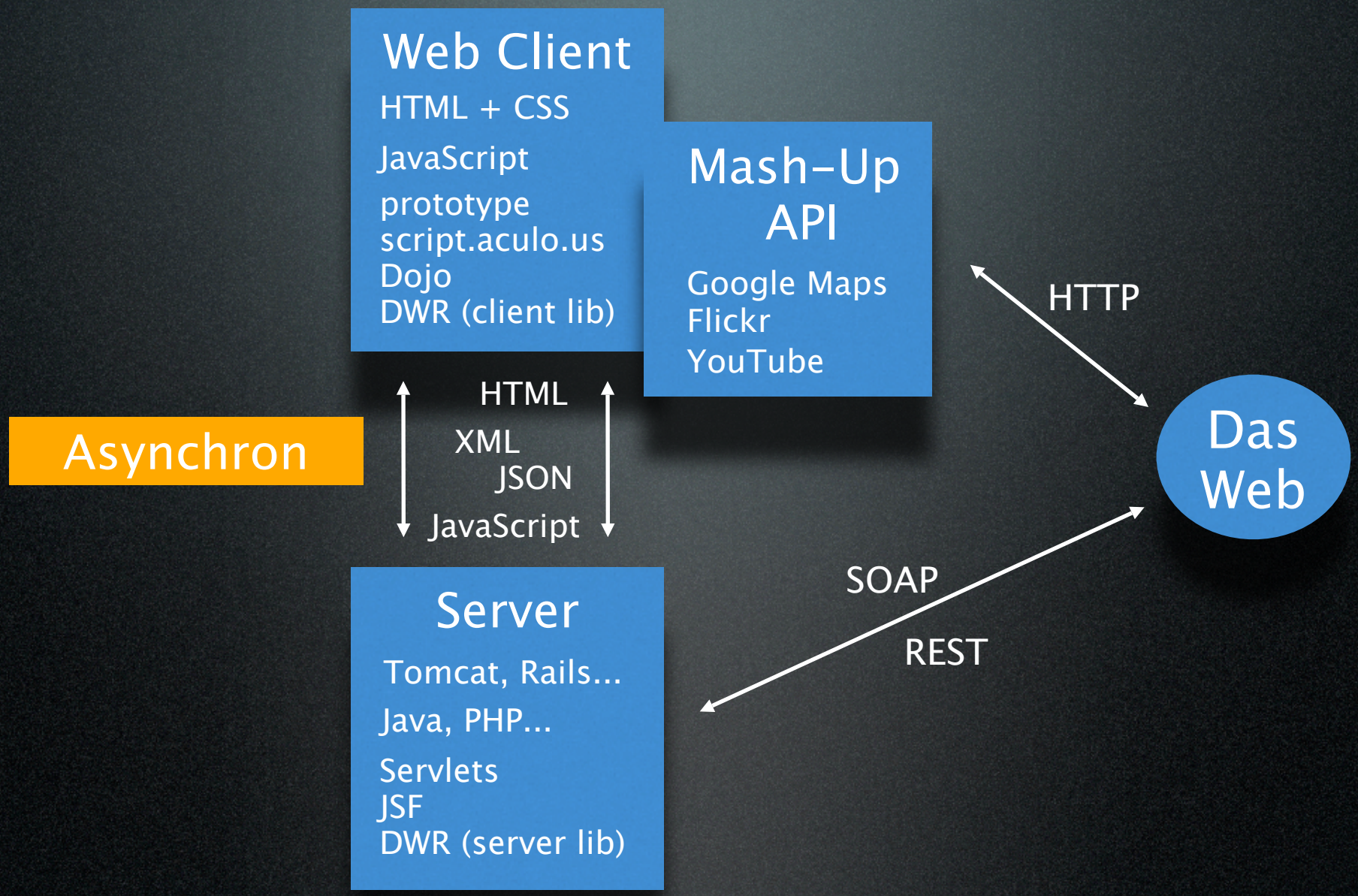
Tomcat, Rails...

Java, PHP...

Servlets
JSF
DWR (server lib)


Web Client
HTML + CSS
JavaScript
prototype
script.aculo.us
Dojo
DWR (client lib)
Asynchron
HTML
XML
JSON
JavaScript
Server
Tomcat, Rails...
Java, PHP...
Servlets
JSF
DWR (server lib)


Web Client
HTML + CSS
JavaScript
prototype
script.aculo.us
Dojo
DWR (client lib)
Mash-Up API
Google Maps
Flickr
YouTube
HTTP
Das Web
Asynchron
HTML
XML
JSON
JavaScript
Server
Tomcat, Rails...
Java, PHP...
Servlets
JSF
DWR (server lib)


Web Client
HTML + CSS
JavaScript
prototype
script.aculo.us
Dojo
DWR (client lib)
Mash-Up API
Google Maps
Flickr
YouTube
HTTP
Das Web
Asynchron
HTML
XML
JSON
JavaScript
SOAP
REST
Server
Tomcat, Rails...
Java, PHP...
Servlets
JSF
DWR (server lib)

Herausforderungen

Herausforderungen

- Technologie-Mix (JavaScript, Java, ...)

Herausforderungen

- Technologie-Mix (JavaScript, Java, ...)
- Verteilung

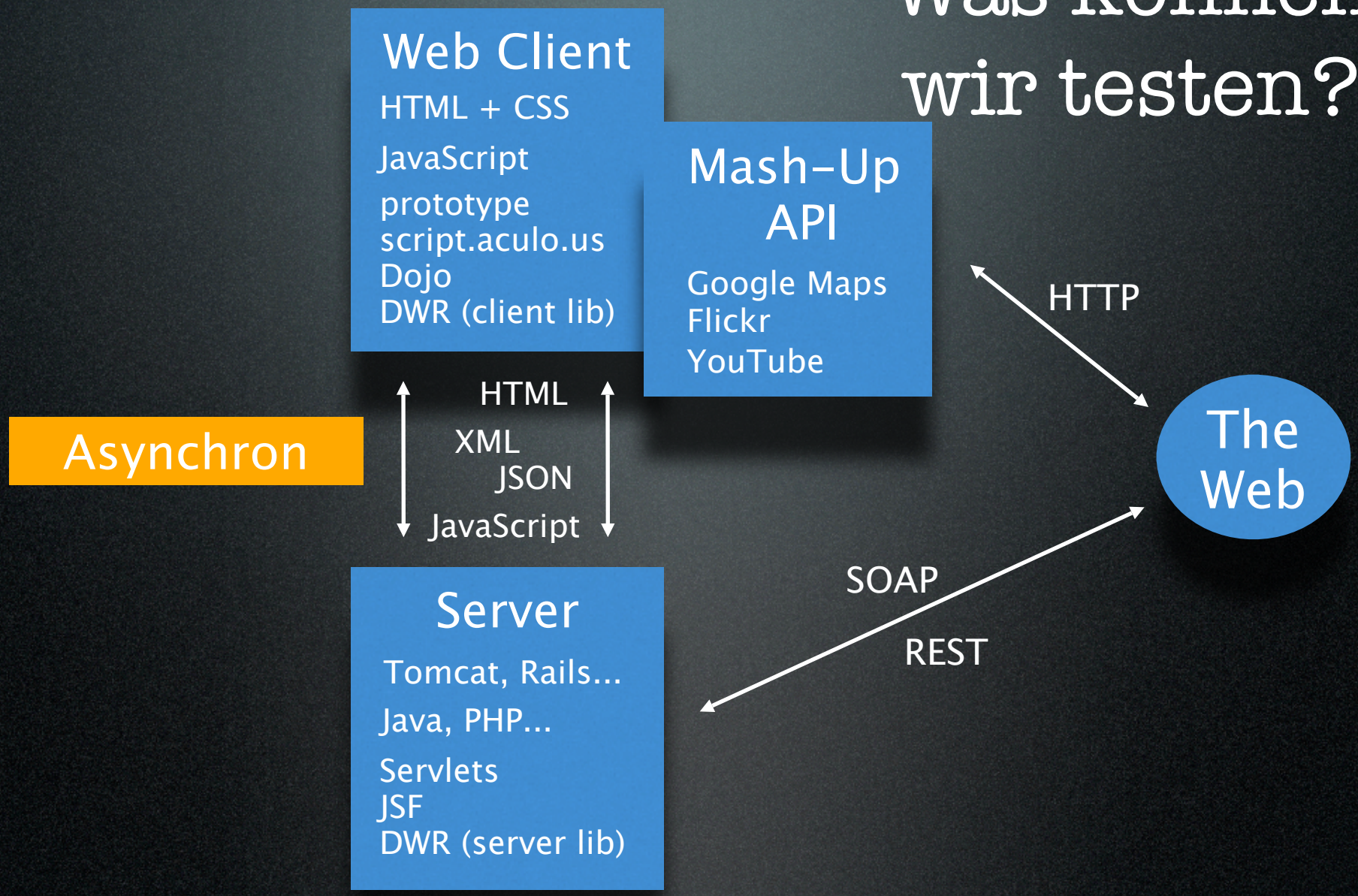
Herausforderungen

- Technologie-Mix (JavaScript, Java, ...)
- Verteilung
- Externe Komponenten
  - über Mash-Ups (z.B. Google Maps)
  - Server-seitige „Remote Services“

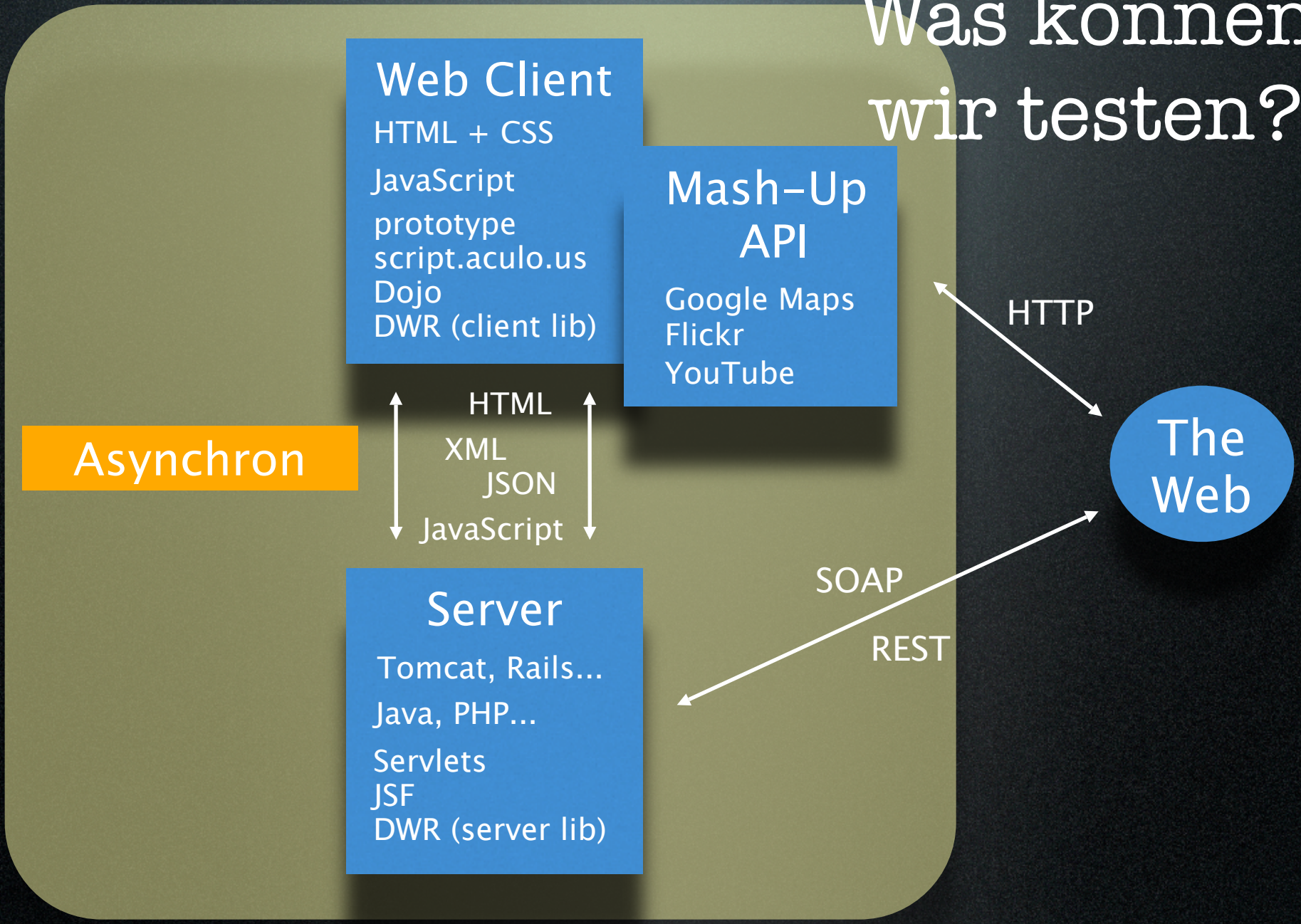
Herausforderungen

- Technologie-Mix (JavaScript, Java, ...)
- Verteilung
- Externe Komponenten
  - über Mash-Ups (z.B. Google Maps)
  - Server-seitige „Remote Services“
- Browser-Inkompatibilitäten

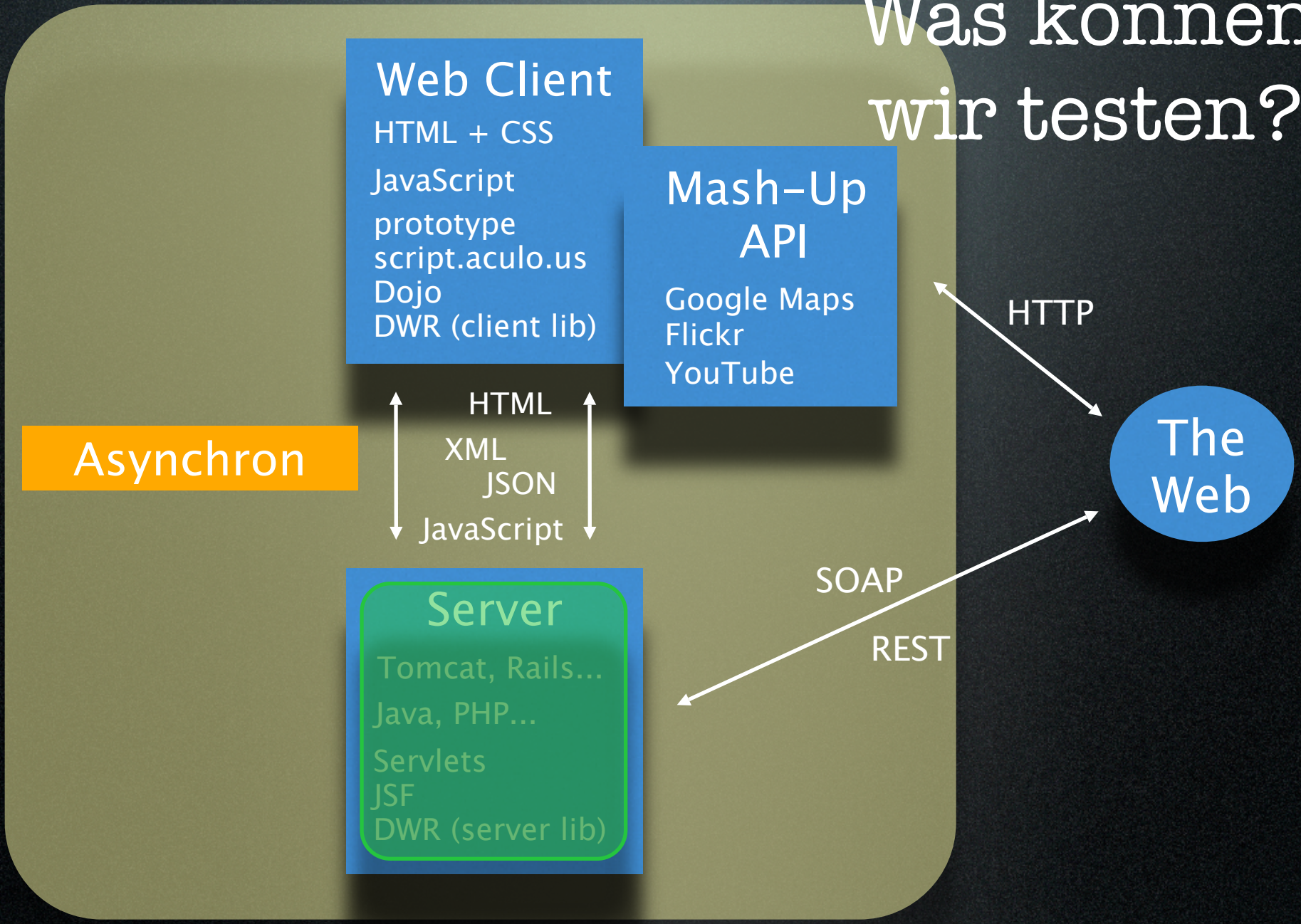
Was können wir testen?



Was können wir testen?



Was können wir testen?



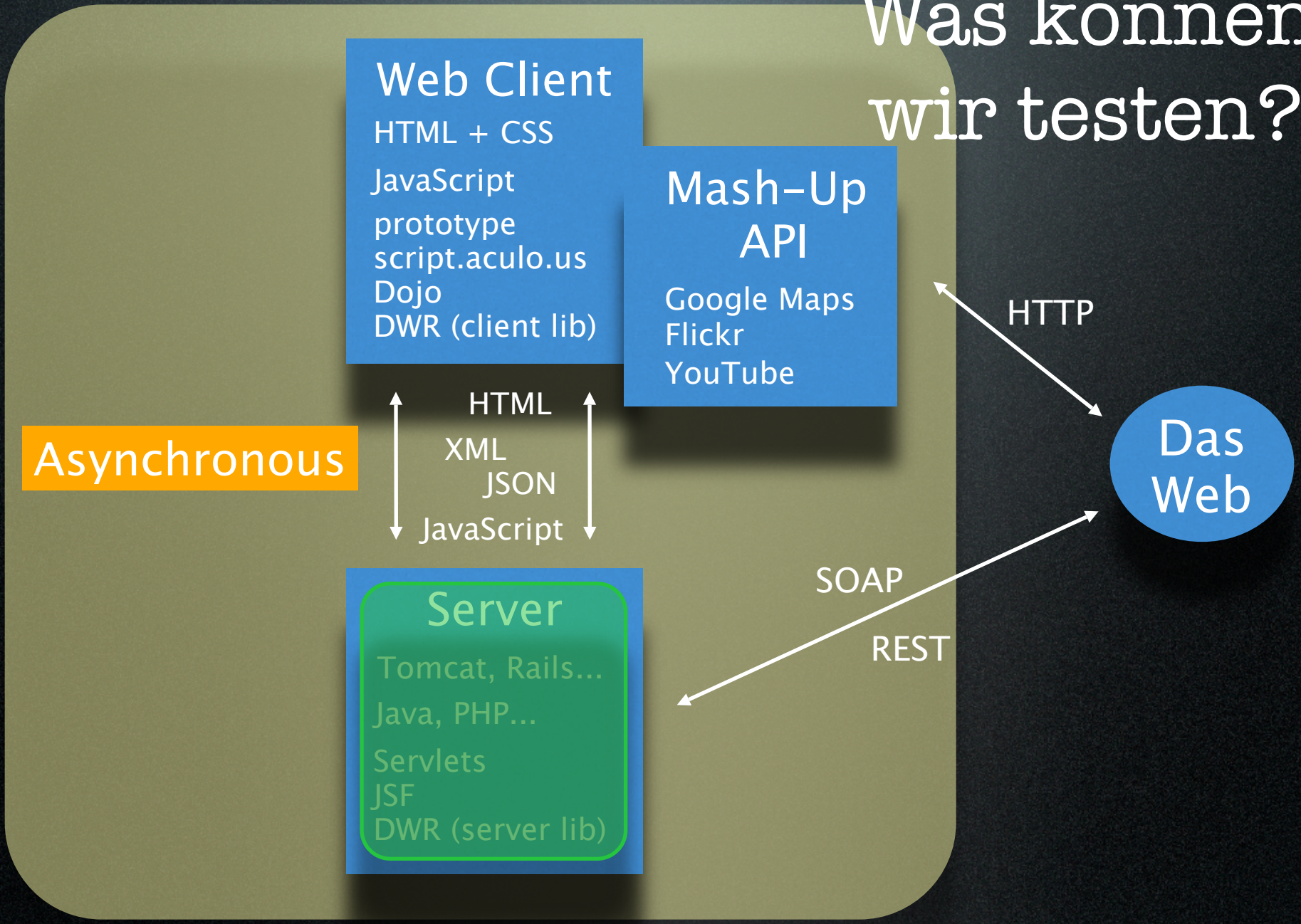
Entwicklertests auf dem Server

Wie gehabt:

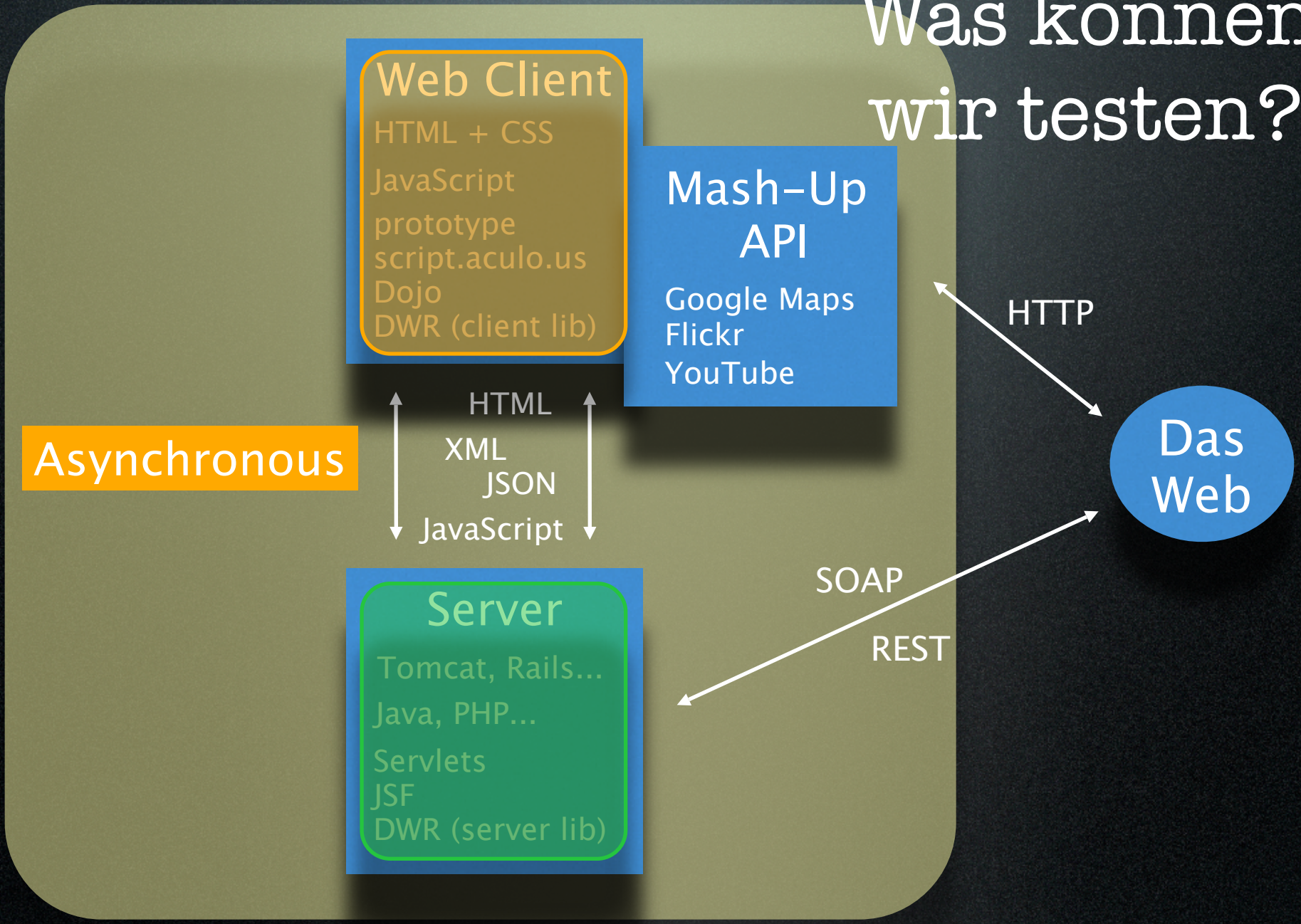
Unit Tests...

... und auch ein paar Integrationstests

Was können wir testen?



Was können wir testen?



Entwicklertests für den Client

Entwicklertests für den Client

- Testgetriebenes JavaScript ist möglich
- Benutze eine Cross-Browser-Bibliothek
- Organisiere deinen Code von Anfang an
- Die Werkzeuge...

Scriptaculous Testing Framework

- http://github.com/madrobby/scriptaculous/wikis/unit-testing
- Testfälle auf HTML-Seiten
- Vorteile
  - Browser-Inkompatibilitäten werden sichtbar
  - Browser-Features sind verfügbar
- Nachteile
  - Automatisierung benötigt auch einen Browser (oder eine Simulation)
  - Abhängig von prototype

```
<head>
  <script src="/js/prototype.js" type="text/javascript"></script>
  <script src="/js/scriptaculous/scriptaculous.js?load=unittest"
          type="text/javascript"></script>
</head>
<body>
  <!-- Log output -->
  <div id="testlog"></div>
  <!-- Tests follow -->
  <script type="text/javascript" language="javascript">
// <![CDATA[

  var myAdder = function (a, b) {...}

  new Test.Unit.Runner({
    setup: function() {
      one = 1;
      two = 2;
    },
    testAddSmallNumbers: function() { with(this) {
      assertEqual(3, myAdder(one, two));
    }},
    ...
});
// ]]>
  </script></body>
```

```
<head>
  <script src="/js/prototype.js" type="text/javascript"></script>
  <script src="/js/scriptaculous/scriptaculous.js?load=unittest"
          type="text/javascript"></script>
</head>
<body>
  <!-- Log output -->
  <div id="testlog"></div>
  <!-- Tests follow -->
  <script type="text/javascript" language="javascript">
// <![CDATA[

  var myAdder = function (a, b) {...}

  new Test.Unit.Runner({
    setup: function() {
      one = 1;
      two = 2;
    },
    testAddSmallNumbers: function() { with(this) {
      assertEqual(3, myAdder(one, two));
    }},
    ...
});
// ]]>
  </script></body>
```

```
<head>
  <script src="/js/prototype.js" type="text/javascript"></script>
  <script src="/js/scriptaculous/scriptaculous.js?load=unittest"
          type="text/javascript"></script>
</head>
<body>
  <!-- Log output -->
  <div id="testlog"></div>
  <!-- Tests follow -->
  <script type="text/javascript" language="javascript">
// <![CDATA[

  var myAdder = function (a, b) {...}

  new Test.Unit.Runner({
    setup: function() {
      one = 1;
      two = 2;
    },
    testAddSmallNumbers: function() { with(this) {
      assertEqual(3, myAdder(one, two));
    }},
    ...
});
// ]]>
  </script></body>
```

```
<head>
  <script src="/js/prototype.js" type="text/javascript"></script>
  <script src="/js/scriptaculous/scriptaculous.js?load=unittest"
          type="text/javascript"></script>
</head>
<body>
  <!-- Log output -->
  <div id="testlog"></div>
  <!-- Tests follow -->
  <script type="text/javascript" language="javascript">
// <![CDATA[

  var myAdder = function (a, b) {...}

  new Test.Unit.Runner({
    setup: function() {
      one = 1;
      two = 2;
    },
    testAddSmallNumbers: function() { with(this) {
      assertEqual(3, myAdder(one, two));
    }},
    ...
});
// ]]>
  </script></body>
```

```
<head>
  <script src="/js/prototype.js" type="text/javascript"></script>
  <script src="/js/scriptaculous/scriptaculous.js?load=unittest"
          type="text/javascript"></script>
</head>
<body>
  <!-- Log output -->
  <div id="testlog"></div>
  <!-- Tests follow -->
  <script type="text/javascript" language="javascript">
// <![CDATA[

  var myAdder = function (a, b) {...}

  new Test.Unit.Runner({
    setup: function() {
      one = 1;
      two = 2;
    },
    testAddSmallNumbers: function() { with(this) {
      assertEqual(3, myAdder(one, two));
    }},
    ...
});
// ]]>
  </script></body>
```

```
<head>
  <script src="/js/prototype.js" type="text/javascript"></script>
  <script src="/js/scriptaculous/scriptaculous.js?load=unittest"
          type="text/javascript"></script>
</head>
```

3 tests, 2 assertions, 1 failures, 0 errors

Status	Test	Message
passed	testAddSmallNumbers	1 assertions, 0 failures, 0 errors
passed	testAddBigNumbers	1 assertions, 0 failures, 0 errors
failed	testAddStringNumbers	0 assertions, 1 failures, 0 errors Failure: assertEqual: expected "'3'", actual "'12'"

```
  new Test.Unit.Runner({
    setup: function() {
      one = 1;
      two = 2;
    },
    testAddSmallNumbers: function() { with(this) {
      assertEqual(3, myAdder(one, two));
    }},
    ...
});
// ]]>
  </script></body>
```

JsTester

- http://jstester.sourceforge.net/
- Benutzt Rhino-JavaScript-Engine um Tests auf dem Server auszuführen
- Vorteile
  - Kein Browser notwendig
  - Einfache Integration mit JUnit / TestNG
- Nachteile
  - Die Funktionalität des Browsers ist nicht verfügbar
  - Browser-Probleme werden nicht entdeckt
- Besonders wertvoll, wenn der Server selbst JavaScript-Code (oder JSON) generiert

MockMe:

http://johanneslink.net/projects/mockme.html

Object to mock:

```
var Speaker = {
  say: function(msg) {
    alert(msg);
  }
};
```

Object under test:

```
var DoubleSpeaker = {
  say: function(msg) {
    Speaker.say(msg+msg);
  }
};
```

MockMe:

http://johanneslink.net/projects/mockme.html

Object to mock:

```
var Speaker = {
  say: function(msg) {
    alert(msg);
  }
};
```

Object under test:

```
var DoubleSpeaker = {
  say: function(msg) {
    Speaker.say(msg+msg);
  }
};
```

Unit Test:

```
testDoubleSpeaker: function() {
  mock(Speaker).andDo(function() {
    DoubleSpeaker.say('oops');
    verify(Speaker.say)('oopsoops');
  });
}
```

MockMe:

http://johanneslink.net/projects/mockme.html

Object to mock:

```
var Speaker = {
  say: function(msg) {
    alert(msg);
  }
};
```

Object under test:

```
var DoubleSpeaker = {
  say: function(msg) {
    Speaker.say(msg+msg);
  }
};
```

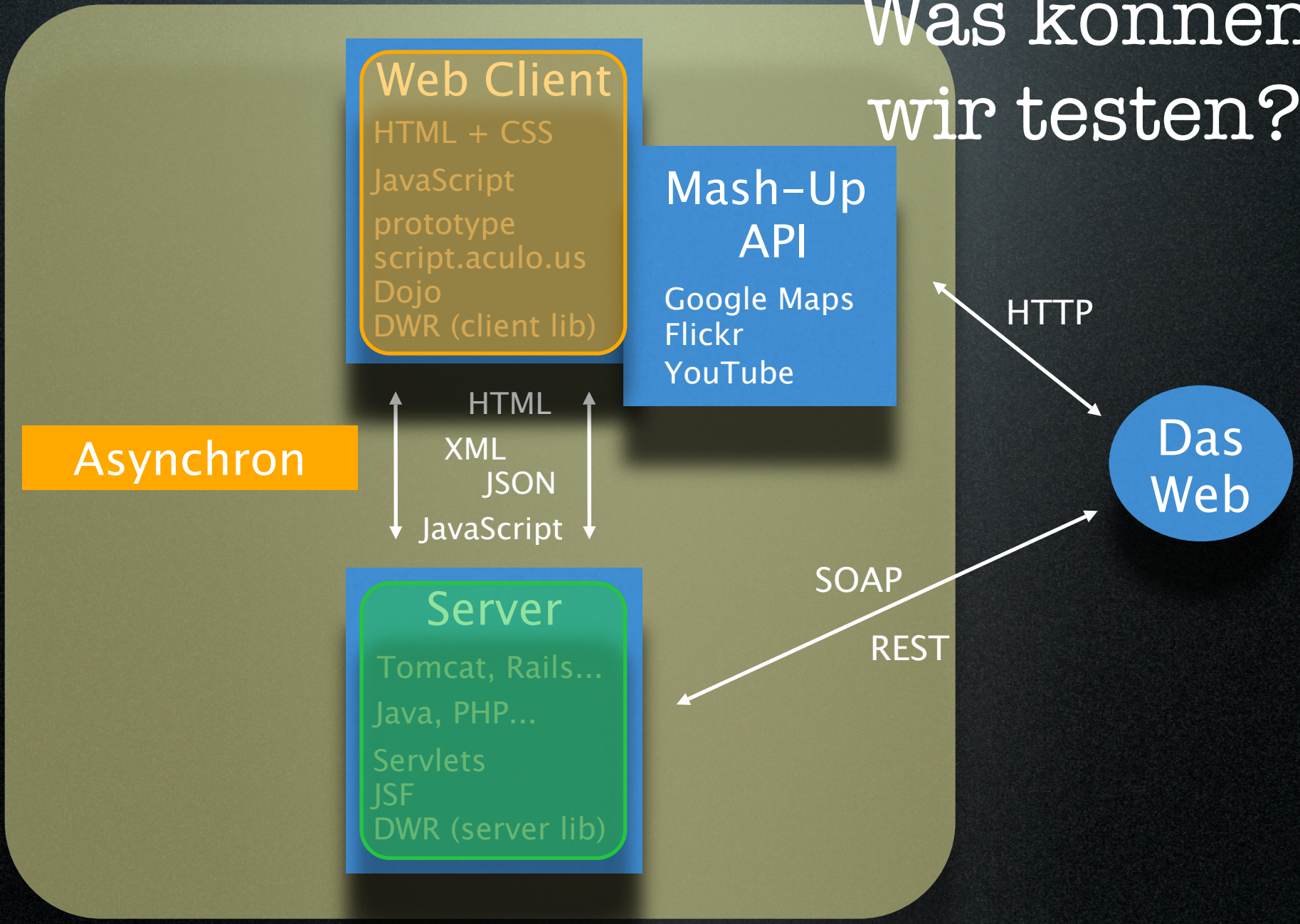
Unit Test:

```
testDoubleSpeaker: function() {
  mock(Speaker).andDo(function() {
    DoubleSpeaker.say('oops');
    verify(Speaker.say)('oopsoops');
  });
}
```

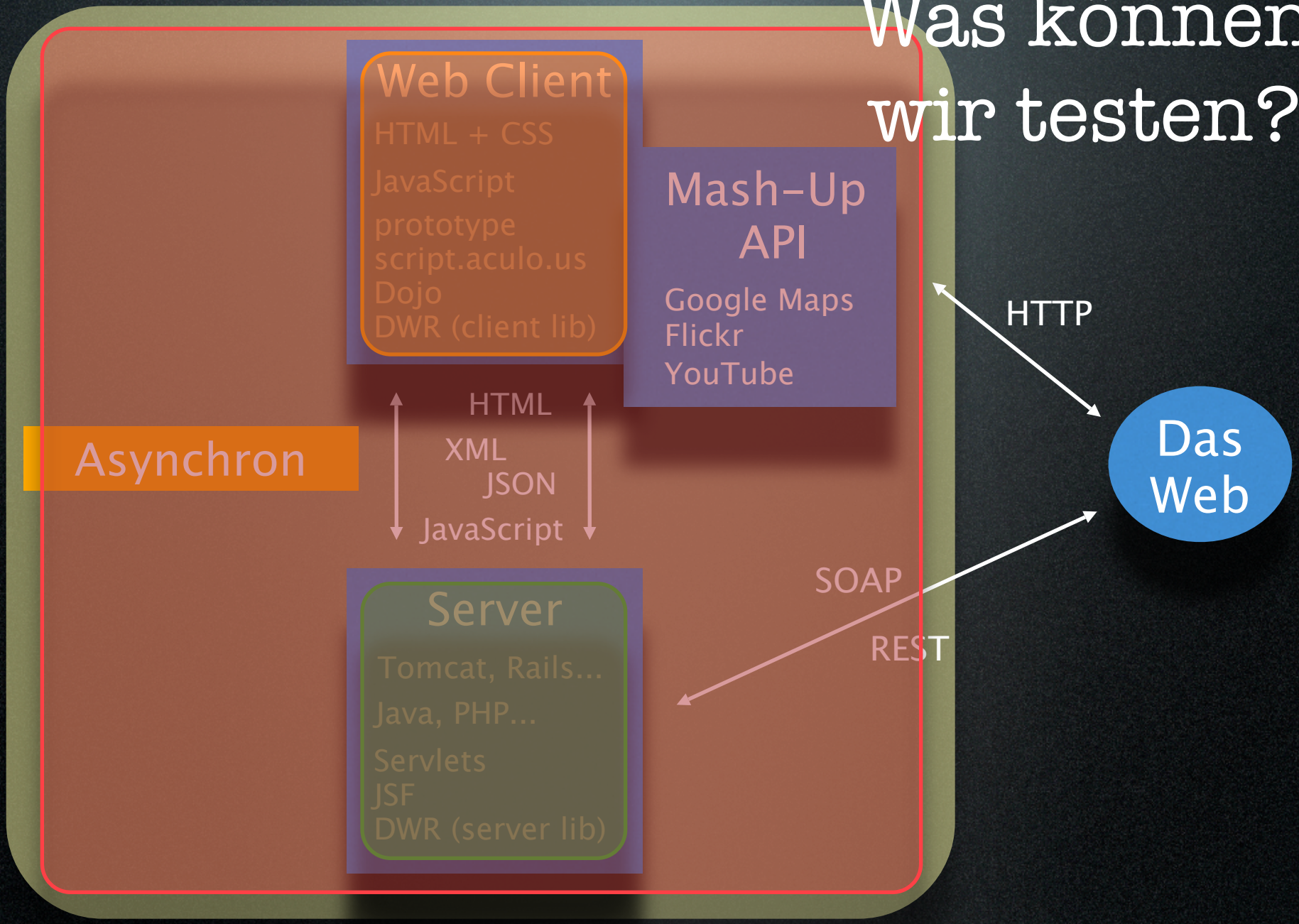
Stubbing:

```
testStubbing: function() { with(this) {
  useMockerFor(function(mocker) {
    var f = mocker.mock();
    when(f)(1, 'two').thenReturn('hiho');
    assertEqual('hiho', f(1, 'two'));
  });
}}
```

Was können wir testen?



Was können wir testen?



Akzeptanztests

- Teste (überwiegend) das ganze System
- Welchen Web-Client?
  - Der echte Browser: Selenium, Watir...
  - Simulierter Browser: HtmlUnit, Webtest, ...
  - „Nur“ die Business-Fassade
- Mocke oder simuliere(?) alle Zugriffe auf externe Komponenten

Ansätze für ATs

- Oberflächenorientiert
  - Tests „bedienen“ die tatsächliche Benutzerschnittstelle
- Fachlich orientiert
  - Formulierung der Testfälle in domänen-naher Sprache

Oberflächenorientierte Akzeptanztests

- Selenium
  - http://selenium.openqa.org/
  - Echter Browser als Ausführungsplattform
  - Selenium IDE oder Remote Control
- Canoo WebTest
  - http://webtest.canoo.com/
  - Verwendet HtmlUnit im Hintergrund
  - Ant-Skripte oder Groovy
- Beide unterstützen JavaScript & asynchrone Aufrufe

Test Suite

Untitled

Untitled		
open	/elo/	
type	newLocation	Toronto
click	addLocationButton	
type	newLocation	
click	eventName	
click	eventDate	
click	publishEventButton	
click	//div[@id='browseEventsMenuItem']/button	

Selenium TestRunner

Execute Tests

Fast — Slow

☐ Highlight elements

Elapsed: 00:00

Tests	Commands
1 run	0 passed
0 failed	0 failed
	0 incomplete

Tools

View DOM | Show Log

Event Location Optimizer

Choose your event location wisely!

My Event | Browse Events

Event Name:

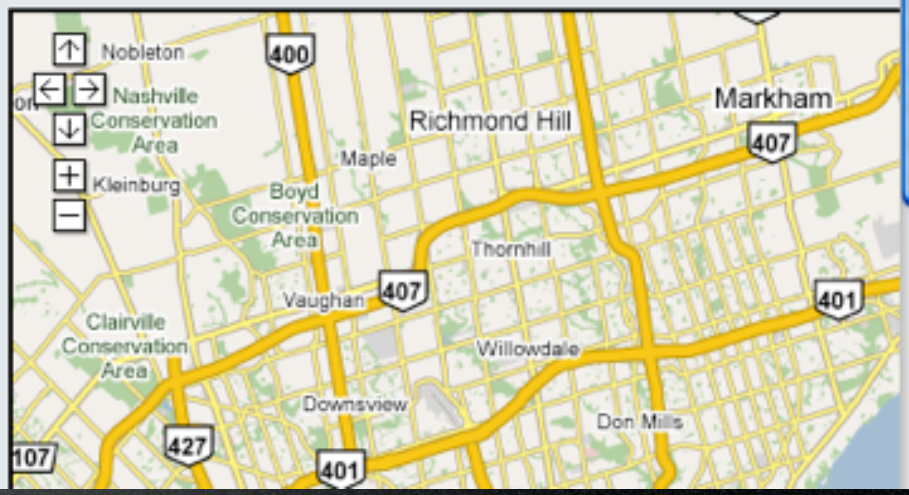
Event Date:

Publish | Reset

New participant from | Add

Participants come from:

Toronto, ON, Canada (2)



Canoo WebTest: Groovy

```
def config_map = [host:"localhost", port:"54321",
                  protocol:"http"]

def ant = new AntBuilder()
ant.taskdef(resource:'webtest.taskdef'){
    classpath(){
        pathelement(location:"$webtest_home/lib")
        fileset(dir:"$webtest_home/lib", includes:"**/*.jar")
    }
}

ant.testSpec(name:'groovy'){
    config(config_map)
    steps(){
        invoke(url:'/plo')
        setInputField(description: 'set a new location',
                htmlId: 'descriptionField',
                value: 'Heidelberg')
        clickButton (description: 'adding...',
                htmlId: 'addEntryButton')
        verifyText(description: 'Check',
                text: 'Heidelberg')
    }
}
```

Selenium:
Vor- und Nachteile

Selenium: Vor- und Nachteile

+ Der echte Browser kommt zum Einsatz

Selenium: Vor- und Nachteile

+ Der echte Browser kommt zum Einsatz

- Den echten Browser zu benutzen ist langsam

Selenium: Vor- und Nachteile

+ Der echte Browser kommt zum Einsatz

- Den echten Browser zu benutzen ist langsam

+ Man muss sich über die Testsprache keine Gedanken machen

Selenium: Vor- und Nachteile

+ Der echte Browser kommt zum Einsatz

- Den echten Browser zu benutzen ist langsam

+ Man muss sich über die Testsprache keine Gedanken machen

- Vielleicht mag man die vorgegebene Testsprachen nicht

Selenium: Vor- und Nachteile

+ Der echte Browser kommt zum Einsatz

- Den echten Browser zu benutzen ist langsam

+ Man muss sich über die Testsprache keine Gedanken machen

- Vielleicht mag man die vorgegebene Testsprachen nicht

- Tests tendieren zu Duplikation

Selenium: Vor- und Nachteile

+ Der echte Browser kommt zum Einsatz

- Den echten Browser zu benutzen ist langsam

+ Man muss sich über die Testsprache keine Gedanken machen

- Vielleicht mag man die vorgegebene Testsprachen nicht

- Tests tendieren zu Duplikation

- Aufwändige „Backdoor“-Mechanismen

Fachlich orientiertes Framework: FIT / FitNesse

- Framework for Integrated Tests
- Testdaten und Skripte in Tabellen (HTML, Excel or Wiki)
- Zielt auf die Sprache des Kunden
- Anbindung ans System mit Java, C#, Python, C++

Test Results

- Test
- Edit
- Versions
- Properties
- Refactor
- Where Used
- RecentChanges
- Files
- Search

Assertions: 24 right, 0 wrong, 0 ignored, 0 exceptions

▶ *Set Up: .EventLocationOptimizer.StoryTestsOnFirefox.SuiteSetUp*

▶ *Set Up: .EventLocationOptimizer.StoryTestsOnFirefox.SetUp*

reset ELO

start fresh

set event name	Agile 2008	and date	2008-08-06
publish current event			

set event name	Golden Wedding	and date	2056-07-21
publish current event			

set event name	My Fortieth Birthday	and date	2009-01-20
publish current event			

Events are sorted by date

check published events	
name	date
Agile 2008	2008-08-06
My Fortieth Birthday	2009-01-20
Golden Wedding	2056-07-21

FIT: Vor- & Nachteile

+ Tests sind von der Entwicklung entkoppelt

+ Testsprache kann frei definiert werden

- Erfordert (ein wenig) Programmierung

- Die passende Testsprache zu finden ist nicht einfach

Wie verbinden wir FIT mit Ajax?

Wie verbinden wir FIT mit Ajax?

Option 1:

Verwende die Business-Facade auf dem Server

Wie verbinden wir FIT mit Ajax?

Option 1:

Verwende die Business-Facade auf dem Server

Option 2:

Verwende einen Web-Client

- Selenium Remote Control
- HtmlUnit

Wie verbinden wir FIT mit Ajax?

Option 1:

Verwende die Business-Facade auf dem Server

Option 2:

Verwende einen Web-Client

- Selenium Remote Control
- HtmlUnit

Option 3:

Verwende eine generische Web-Fixture

- http://fitnesse.info/webtest/
- http://htmlfixtureim.sourceforge.net/

Event Location Optimizer

Event Location Optimizer

Event Location Optimizer

Choose your event location wisely!

My Event | Browse Events

Event Name: Agile 2008

Event Date: 2008-08-05

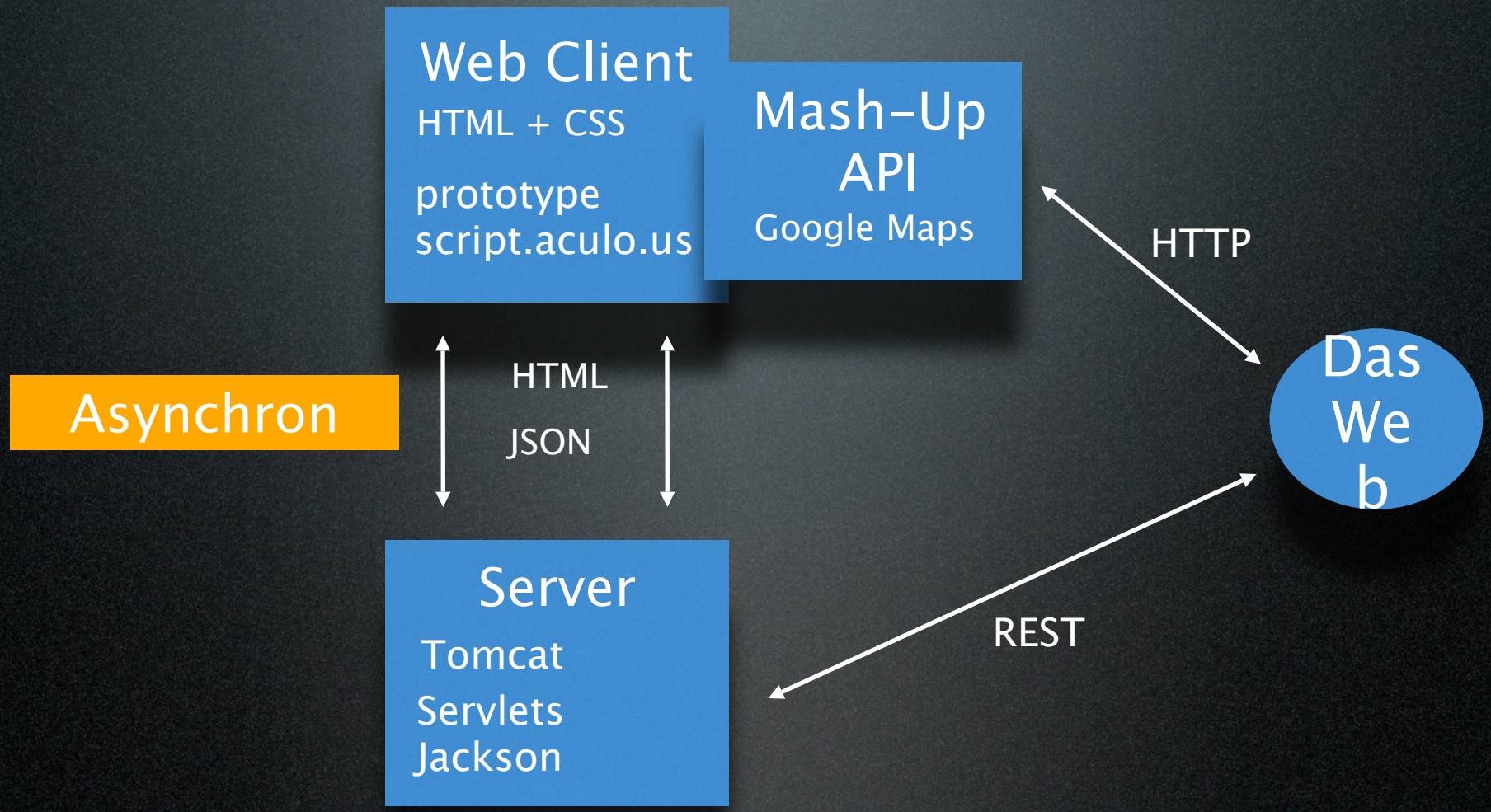
Publish | Reset

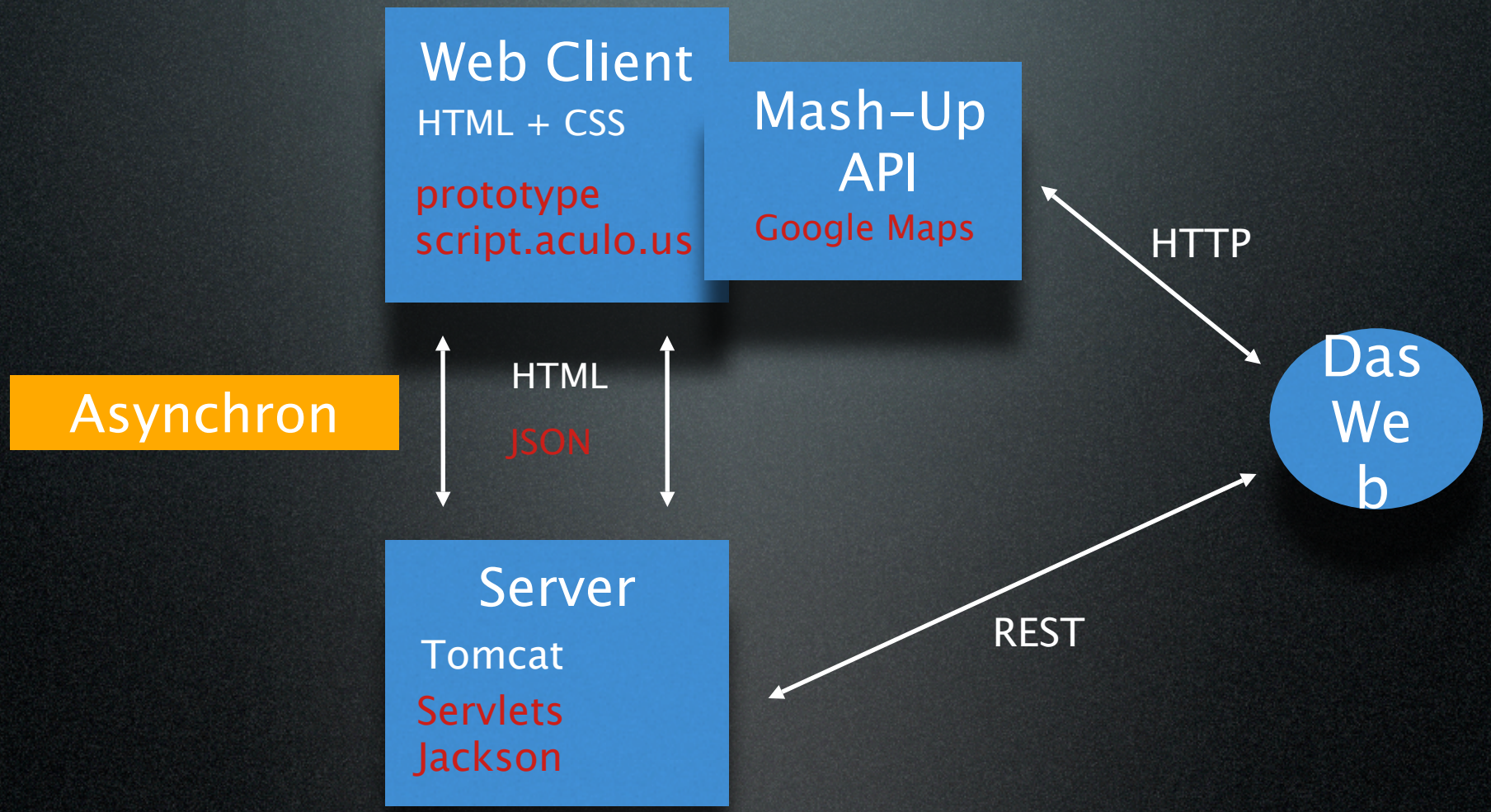
New participant from los angeles Add

Participants come from:

Heidelberg, Germany
San Francisco, CA, USA
Los Angeles, CA, USA

ELO Demo


Web Client
HTML + CSS
prototype
script.aculo.us
Mash-Up
API
Google Maps
HTTP
Asynchron
HTML
JSON
Das
We
b
Server
Tomcat
Servlets
Jackson
REST


Web Client
HTML + CSS
prototype
script.aculo.us
Mash-Up
API
Google Maps
HTTP
Asynchron
HTML
JSON
Das
We
b
Server
Tomcat
Servlets
Jackson
REST

ELO: JS Unit Tests

- TDD der Logik ist einfach
- Aber wie testet man die ganze DOM-Manipulation?

Unit Testing
DOM Manipulation (1)

Unit Testing
DOM Manipulation (1)

```
<div id="browseEventsMenuItem"><button>Browse Events</button></div>
<div id="myEventMenuItem" class="selected"><button>My Event</button></div>
<div id="browseEventsScreen">Browse Events Screen</div>
<div id="myEventScreen" class="showing">My Event Screen</div>
```

Unit Testing
DOM Manipulation (1)

```
<div id="browseEventsMenuItem"><button>Browse Events</button></div>
<div id="myEventMenuItem" class="selected"><button>My Event</button></div>
<div id="browseEventsScreen">Browse Events Screen</div>
<div id="myEventScreen" class="showing">My Event Screen</div>
```

```
button {color: black;}

.menuItem.selected button {color: blue;}

.screen {display: none;}

.screen.showing {display: inherit;}
```

Unit Testing
DOM Manipulation (1)

```
<div id="browseEventsMenuItem"><button>Browse Events</button></div>
<div id="myEventMenuItem" class="selected"><button>My Event</button></div>
<div id="browseEventsScreen">Browse Events Screen</div>
<div id="myEventScreen" class="showing">My Event Screen</div>
```

```
testBrowseEventsMenuButton: function() { with(this) {
   browseEventsButtonClicked();
   assert(Element.hasClassName($('browseEventsMenuItem'), ELO.SELECTED));
   assert(!Element.hasClassName($('myEventMenuItem'), ELO.SELECTED));
   assert(Element.hasClassName($('browseEventsScreen'), ELO.SHOWING));
   assert(!Element.hasClassName($('myEventScreen'), ELO.SHOWING));
}}
```

Unit Testing DOM Manipulation (1)

```
<div id="browseEventsMenuItem"><button>Browse Events</button></div>
<div id="myEventMenuItem" class="selected"><button>My Event</button></div>
<div id="browseEventsScreen">Browse Events Screen</div>
<div id="myEventScreen" class="showing">My Event Screen</div>
```

```
testBrowseEventsMenuButton: function() { with(this) {
   browseEventsButtonClicked();
   assert(Element.hasClassName($('browseEventsMenuItem'), ELO.SELECTED));
   assert(!Element.hasClassName($('myEventMenuItem'), ELO.SELECTED));
   assert(Element.hasClassName($('browseEventsScreen'), ELO.SHOWING));
   assert(!Element.hasClassName($('myEventScreen'), ELO.SHOWING));
}}
```

```
$("browseEventsMenuItem").observe('click', browseEventsButtonClicked);

function browseEventsButtonClicked() {
   $("myEventMenuItem").removeClassName(ELO.SELECTED);
   $("myEventScreen").removeClassName(ELO.SHOWING);
   $("browseEventsMenuItem").addClassName(ELO.SELECTED);
   $("browseEventsScreen").addClassName(ELO.SHOWING);
}
```

Unit Testing
DOM Manipulation (1)

```
<div id="browseEventsMenuItem"><button>Browse Events</button></div>
<div id="myEventMenuItem" class="selected"><button>My Event</button></div>
<div id="browseEventsScreen">Browse Events Screen</div>
<div id="myEventScreen" class="showing">My Event Screen</div>
```

```
testBrowseEventsMenuButton: function() { with(this) {
    $('browseEventsMenuItem').down('button').click();
    assert(Element.hasClassName($('browseEventsMenuItem'), ELO.SELECTED));
    assert(!Element.hasClassName($('myEventMenuItem'), ELO.SELECTED));
    assert(Element.hasClassName($('browseEventsScreen'), ELO.SHOWING));
    assert(!Element.hasClassName($('myEventScreen'), ELO.SHOWING));
}}
```

```
$("browseEventsMenuItem").observe('click', browseEventsButtonClicked);

function browseEventsButtonClicked() {
    $("myEventMenuItem").removeClassName(ELO.SELECTED);
    $("myEventScreen").removeClassName(ELO.SHOWING);
    $("browseEventsMenuItem").addClassName(ELO.SELECTED);
    $("browseEventsScreen").addClassName(ELO.SHOWING);
}
```

Unit Testing
DOM Manipulation (2)

```
<div id="browseEventsMenuItem"><button>Browse Events</button></div>
<div id="myEventMenuItem" class="selected"><button>My Event</button></div>
<div id="browseEventsScreen">Browse Events Screen</div>
<div id="myEventScreen" class="showing">My Event Screen</div>
```

```
testBrowseEventsMenuButton: function() { with(this) {
   browseEventsButtonClicked();
   ...;
}},

testMyEventMenuButton: function() { with(this) {
   myEventButtonClicked();
   ...
}}
```

Unit Testing
DOM Manipulation (2)

```
<div id="browseEventsMenuItem"><button>Browse Events</button></div>
<div id="myEventMenuItem" class="selected"><button>My Event</button></div>
<div id="browseEventsScreen">Browse Events Screen</div>
<div id="myEventScreen" class="showing">My Event Screen</div>
```

```
setup: function() { with(this) {
   mydom = $('myDom').innerHTML;
}},

testBrowseEventsMenuButton: function() { with(this) {
   browseEventsButtonClicked();
   ...;
}},

testMyEventMenuButton: function() { with(this) {
   myEventButtonClicked();
   ...
}}

teardown: function() { with(this) {
   $('mydom').update(mydom);
}},
```

Unit Testing
Visual Effects (1)

Unit Testing Visual Effects (1)

```
function crossover(from, to) {
    Effect.Fade(from);
    Effect.Appear(to, {queue: 'end'});
}
```

Unit Testing Visual Effects (1)

```
function crossover(from, to) {
    Effect.Fade(from);
    Effect.Appear(to, {queue: 'end'});
}
```

```
.screen {display: none;}
.screen.showing {display: inherit;}
```

Unit Testing Visual Effects (1)

```
function crossover(from, to) {
    Effect.Fade(from);
    Effect.Appear(to, {queue: 'end'});
}
```

```
.screen {display: none;}
.screen.showing {display: inherit;}
```

```
testBrowseEventsMenuButton: function() { with(this) {
   browseEventsButtonClicked();
   ...
   assertNotVisible($('myEventScreen'));
   assertVisible($('browseEventsScreen'));
}}
```

Unit Testing Visual Effects (2)

```
function crossover(from, to) {
  Effect.Fade(from);
  Effect.Appear(to, {queue: 'end'});
}
```

```
testBrowseEventsMenuButton: function() { with(this) {
   browseEventsButtonClicked();
   ...
   assertNotVisible($('myEventScreen'));
   assertVisible($('browseEventsScreen'));
}}
```

Unit Testing Visual Effects (2)

```
function crossover(from, to) {
  Effect.Fade(from);
  Effect.Appear(to, {queue: 'end'});
}
```

```
testBrowseEventsMenuButton: function() { with(this) {
   browseEventsButtonClicked();
   ...
   assertNotVisible($('myEventScreen'));
   assertVisible($('browseEventsScreen'));
}}
```

Unit Testing Visual Effects (2)

```
function crossover(from, to) {
  Effect.Fade(from);
  Effect.Appear(to, {queue: 'end'});
}
```

```
testBrowseEventsMenuButton: function() { with(this) {
   browseEventsButtonClicked();
   ...
   assertNotVisible($('myEventScreen'));
   assertVisible($('browseEventsScreen'));
}}
```

Unit Testing Visual Effects (2)

```
function crossover(from, to) {
  Effect.Fade(from);
  Effect.Appear(to, {queue: 'end'});
}
```

```
testBrowseEventsMenuButton: function() { with(this) {
  browseEventsButtonClicked();
  ...
  assertNotVisible($('myEventScreen'));
  assertVisible($('browseEventsScreen'));
}}
```

```
testBrowseEventsMenuButton: function() { with(this) {
  ...
  wait(2000, function() {
    assertNotVisible($('browseEventsScreen'));
    assertVisible($('myEventScreen'));
  });
}}
```

Unit Testing Visual Effects (3)

```
function crossover(from, to) {
  Effect.Fade(from);
  Effect.Appear(to, {queue: 'end'});
}
```

Unit Testing Visual Effects (3)

```
function crossover(from, to) {
  Effect.Fade(from);
  Effect.Appear(to, {queue: 'end'});
}
```

```
testBrowseEventsMenuButton: function() { with(this) {
    useMockerFor(function(mocker) {
        mocker.within(Effect).mock('Fade', 'Appear');
        browseEventsButtonClicked();
        verify(Effect.Fade)($('myEventScreen'));
        verify(Effect.Appear)($('browseEventsScreen'), {queue: 'end'});
    });
    assert(Element.hasClassName($('browseEventsMenuItem'), ELO.SELECTED));
    assert(!Element.hasClassName($('myEventMenuItem'), ELO.SELECTED));
    assert(Element.hasClassName($('browseEventsScreen'), ELO.SHOWING));
    assert(!Element.hasClassName($('myEventScreen'), ELO.SHOWING));
}},
```

Unit Testing Google Maps

```
testDisplaySimpleMapOnline: function() { with(this) {
  useMockerFor(function (mocker) {
    mocker.mock(GMap2);
    var gmap = {focus: {longitude: 1.234, lattitude: 4.321},
      zoom: 5, hasEventIcon: false};
    ELO.GMap.display(gmap);
    verify(GMap2.prototype.initialize)($('map'));
    verify(GMap2.prototype.setCenter)(new GLatLng(4.321, 1.234), 5);
  });
}}
```

Unit Testing Google Maps

```
<script type="text/javascript"
  src="http://maps.google.com/maps?
  file=api&amp;v=2&amp;key=ABQIAA[...]thcBeqofUC2FaJTlA">
</script>
```

```
testDisplaySimpleMapOnline: function() { with(this) {
  useMockerFor(function (mocker) {
    mocker.mock(GMap2);
    var gmap = {focus: {longitude: 1.234, lattitude: 4.321},
      zoom: 5, hasEventIcon: false};
    ELO.GMap.display(gmap);
    verify(GMap2.prototype.initialize)($('map'));
    verify(GMap2.prototype.setCenter)(new GLatLng(4.321, 1.234), 5);
  });
}}
```

Unit Testing Google Maps

```
var GMap2 = Class.create({
  initialize: function() {},
  setCenter: function() {}
});

var GLatLng = Class.create({
  initialize: function(lattitude, longitude) {
    this.lattitude = lattitude;
    this.longitude = longitude;
  }
});
```

```
testDisplaySimpleMapOnline: function() { with(this) {
  useMockerFor(function (mocker) {
    mocker.mock(GMap2);
    var gmap = {focus: {longitude: 1.234, lattitude: 4.321},
      zoom: 5, hasEventIcon: false};
    ELO.GMap.display(gmap);
    verify(GMap2.prototype.initialize)($('map'));
    verify(GMap2.prototype.setCenter)(new GLatLng(4.321, 1.234), 5);
  });
}}
```

Unit Testing Ajax Facade

```
testSuccessfulRequest: function() { with(this) {
    ELO.AjaxResponses = {myResponse: function(params) {}};
    mock(Ajax, ELO.AjaxResponses).andDo(function() {
        when(Ajax.Request)(ELO.REQUEST_URL, any()).thenDo(function(url, opts) {
            opts.onSuccess(
                {responseJSON: [{name: "myResponse", params: [1, 2]}]});
        });
        optsMatcher = { match: function(opts) {
            return (opts.postBody ==
                Object.toJSON({name: 'myRequest', params: ['p1', 'p2']}));
        }};
        new ELO.AjaxRequest('myRequest', ['p1', 'p2']);
        verify(Ajax.Request)(ELO.REQUEST_URL, optsMatcher);
        verify(ELO.AjaxResponses.myResponse)([1, 2]);
    });
}}
```

Unit Testing Ajax Facade

```
testSuccessfulRequest: function() { with(this) {
    ELO.AjaxResponses = {myResponse: function(params) {}};
    mock(Ajax, ELO.AjaxResponses).andDo(function() {
        when(Ajax.Request)(ELO.REQUEST_URL, any()).thenDo(function(url, opts) {
            opts.onSuccess(
                {responseJSON: [{name: "myResponse", params: [1, 2]}]});
        });
        optsMatcher = { match: function(opts) {
            return (opts.postBody ==
                Object.toJSON({name: 'myRequest', params: ['p1', 'p2']}));
        }};
        new ELO.AjaxRequest('myRequest', ['p1', 'p2']);
        verify(Ajax.Request)(ELO.REQUEST_URL, optsMatcher);
        verify(ELO.AjaxResponses.myResponse)([1, 2]);
    });
}}
```

Unit Testing Ajax Facade

```
testSuccessfulRequest: function() { with(this) {
    ELO.AjaxResponses = {myResponse: function(params) {}};
    mock(Ajax, ELO.AjaxResponses).andDo(function() {
        when(Ajax.Request)(ELO.REQUEST_URL, any()).thenDo(function(url, opts) {
            opts.onSuccess(
                {responseJSON: [{name: "myResponse", params: [1, 2]}]});
        });
        optsMatcher = { match: function(opts) {
            return (opts.postBody ==
                Object.toJSON({name: 'myRequest', params: ['p1', 'p2']}));
        }};
        new ELO.AjaxRequest('myRequest', ['p1', 'p2']);
        verify(Ajax.Request)(ELO.REQUEST_URL, optsMatcher);
        verify(ELO.AjaxResponses.myResponse)([1, 2]);
    });
}}
```

Automatisierung der JavaScript Unit Tests

start selenium server on port	4444

run scriptaculous tests with selenium in browser	Firefox			
url	tests?	assertions?	failures?	errors?
http://localhost:8080/elo/tests/unittests/test_menuItems.html	2	8	0	0
http://localhost:8080/elo/tests/unittests/test_menuItemsRealEffects.html	2	8	0	0
http://localhost:8080/elo/tests/unittests/test_eventName.html	5	13	0	0
http://localhost:8080/elo/tests/unittests/test_eventDate.html	7	11	0	0
http://localhost:8080/elo/tests/unittests/test_addLocationForm.html	4	1	0	0
http://localhost:8080/elo/tests/unittests/test_publishedEvents.html	2	6	0	0
http://localhost:8080/elo/tests/unittests/test_participantAddress.html	6	16	0	0
http://localhost:8080/elo/tests/unittests/test_ajaxRequest.html	4	8	0	0
http://localhost:8080/elo/tests/unittests/test_ajaxResponses.html	5	9	0	0
http://localhost:8080/elo/tests/unittests/test_errorsDisplay.html	5	15	0	0
http://localhost:8080/elo/tests/unittests/test_buttons.html	2	2	0	0
http://localhost:8080/elo/tests/unittests/test_hashToTableRows.html	4	14	0	0
http://localhost:8080/elo/tests/unittests/test_gmap.html	9	24	0	0
tests: 57, assertions: 135, failures: 0, errors: 0				

stop selenium server

Heuristiken für JavaScript Unit Tests

- Entferne alle Abhängikeiten zum Web
- Man benötigt weniger **explizite** Entkopplung in JavaScript als in Java
- Vermeide Mocks, wenn du die API nicht gut kennst!
- Lass deine Unit Tests auf unterschiedlichen JS-Implementierungen laufen (z.B. Browser & Rhino)

ELO: Akzeptanztests

ELO: Akzeptanztests

- Benutze FitNesse für fachlich-orientierte Szenario-Tests

ELO: Akzeptanztests

- Benutze FitNesse für fachlich-orientierte Szenario-Tests
- Verbinde FitNesse mit der Applikation
  - via Selenium RC
  - (via HtmlUnit)
  - direkt über die Business-Facade

ELO: Akzeptanztests

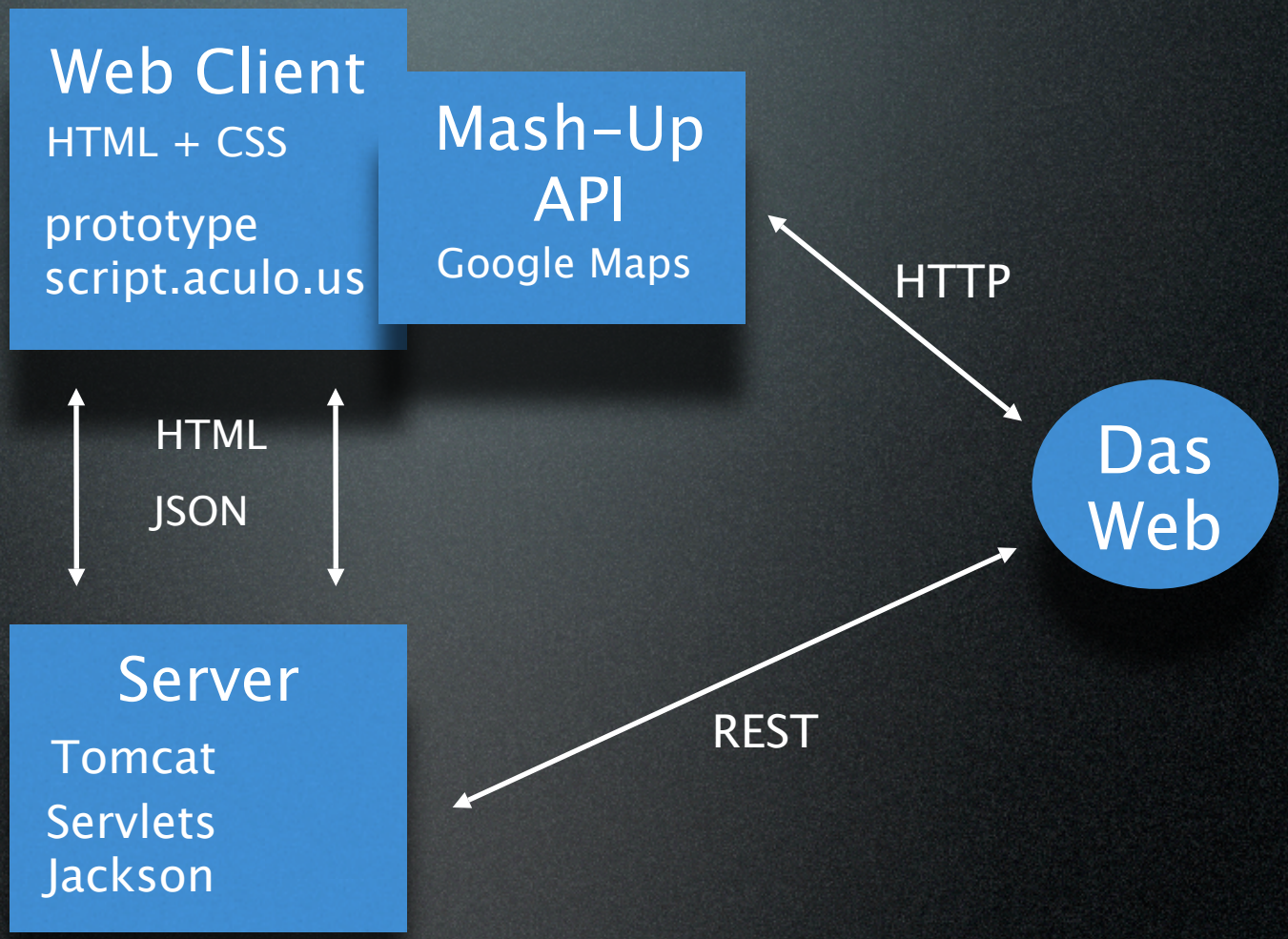
- Benutze FitNesse für fachlich-orientierte Szenario-Tests
- Verbinde FitNesse mit der Applikation
  - via Selenium RC
  - (via HtmlUnit)
  - direkt über die Business-Facade
- Ziel: Mache die Testfälle vom WEB unabhängig

Add participant from Totonto

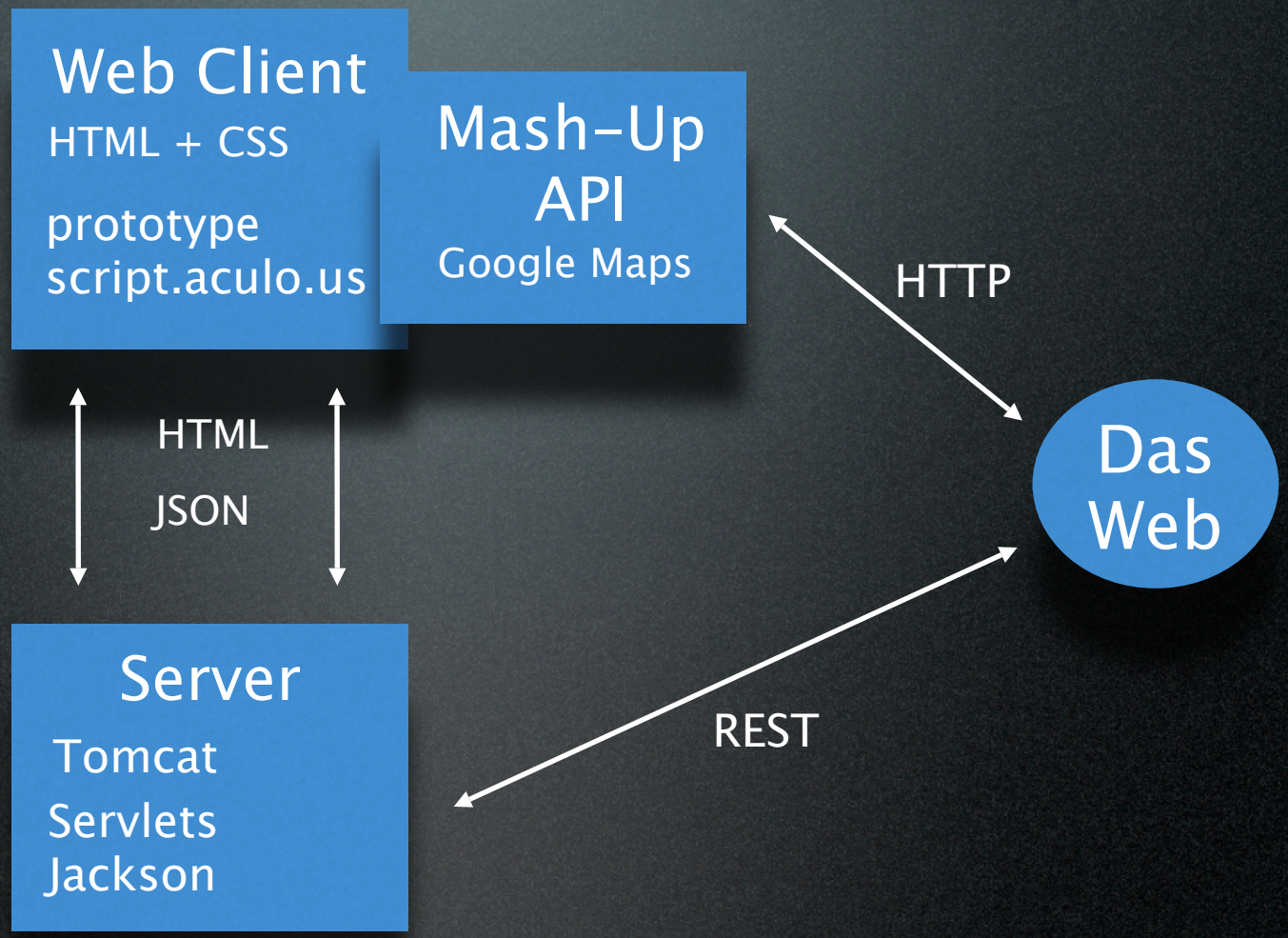
start fresh

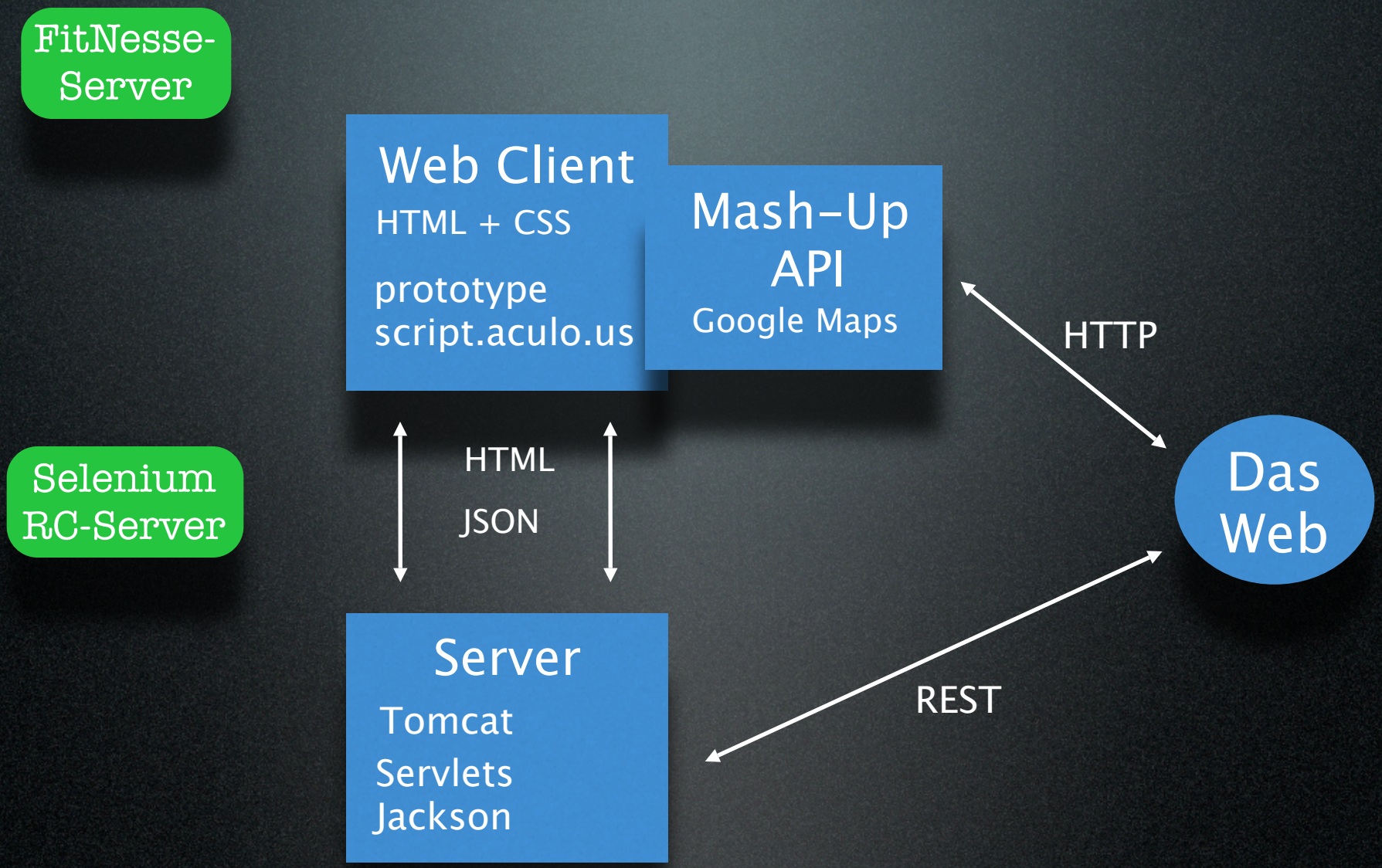
add participant from Toronto

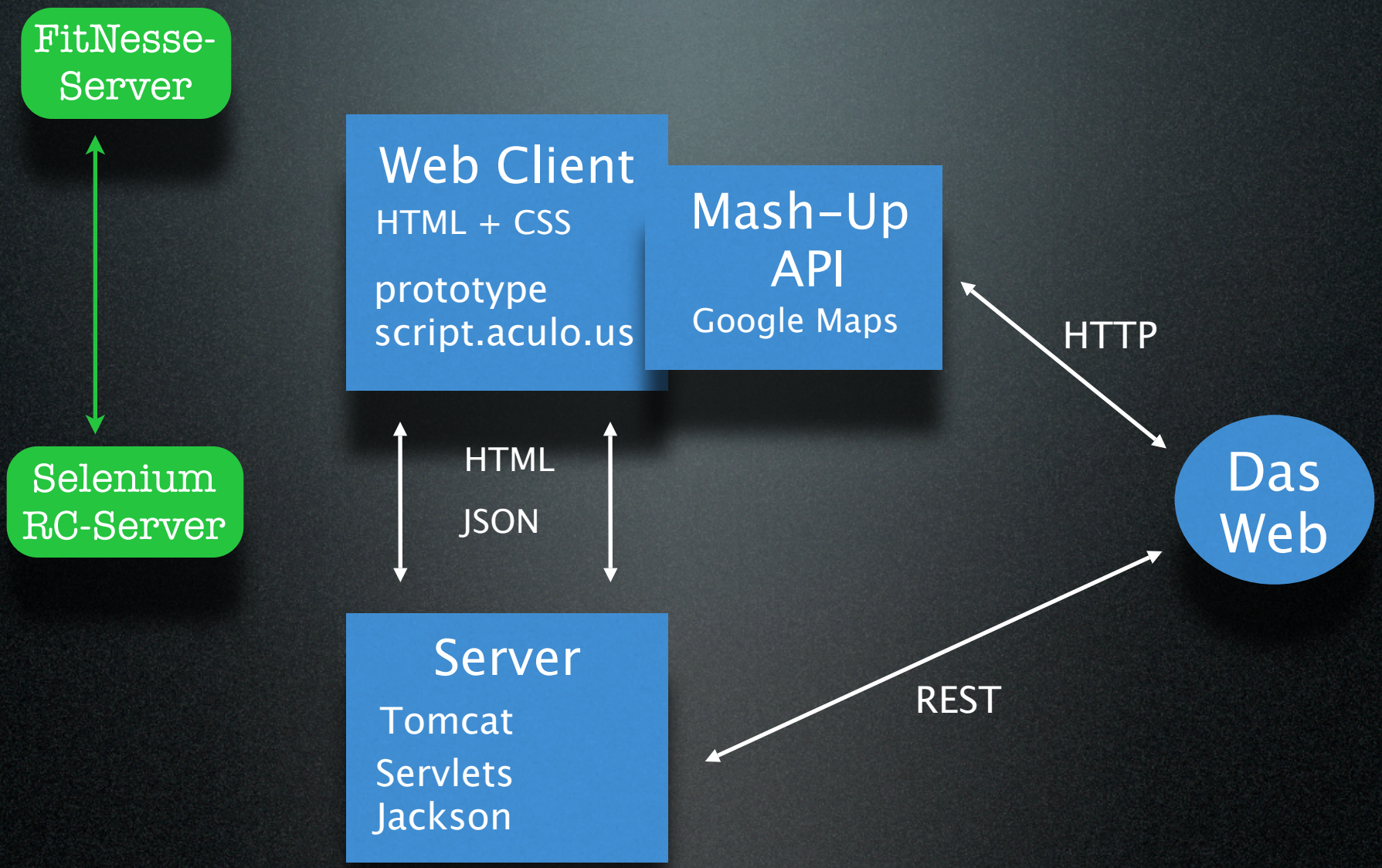

```
public class SeleniumELOFacade implements ELOFacade...

    public boolean startFresh() throws Exception {
        selenium = SeleniumFacade.openBrowser(browser, url);
        selenium.open(url);
        waitUntilIdle();
        return true;
    }

    private void waitUntilIdle() {
        waitForElementToDisappear(IMG_BUSY_INDICATOR);
    }

    public boolean addParticipantFrom(String location) {
        changeScreen(MENU_MY_EVENT);
        selenium.type(INPUT_NEW_LOCATION, location);
        selenium.click(BUTTON_ADD_PARTICIPANT_FROM);
        waitUntilIdle();
        return true;
    }
```

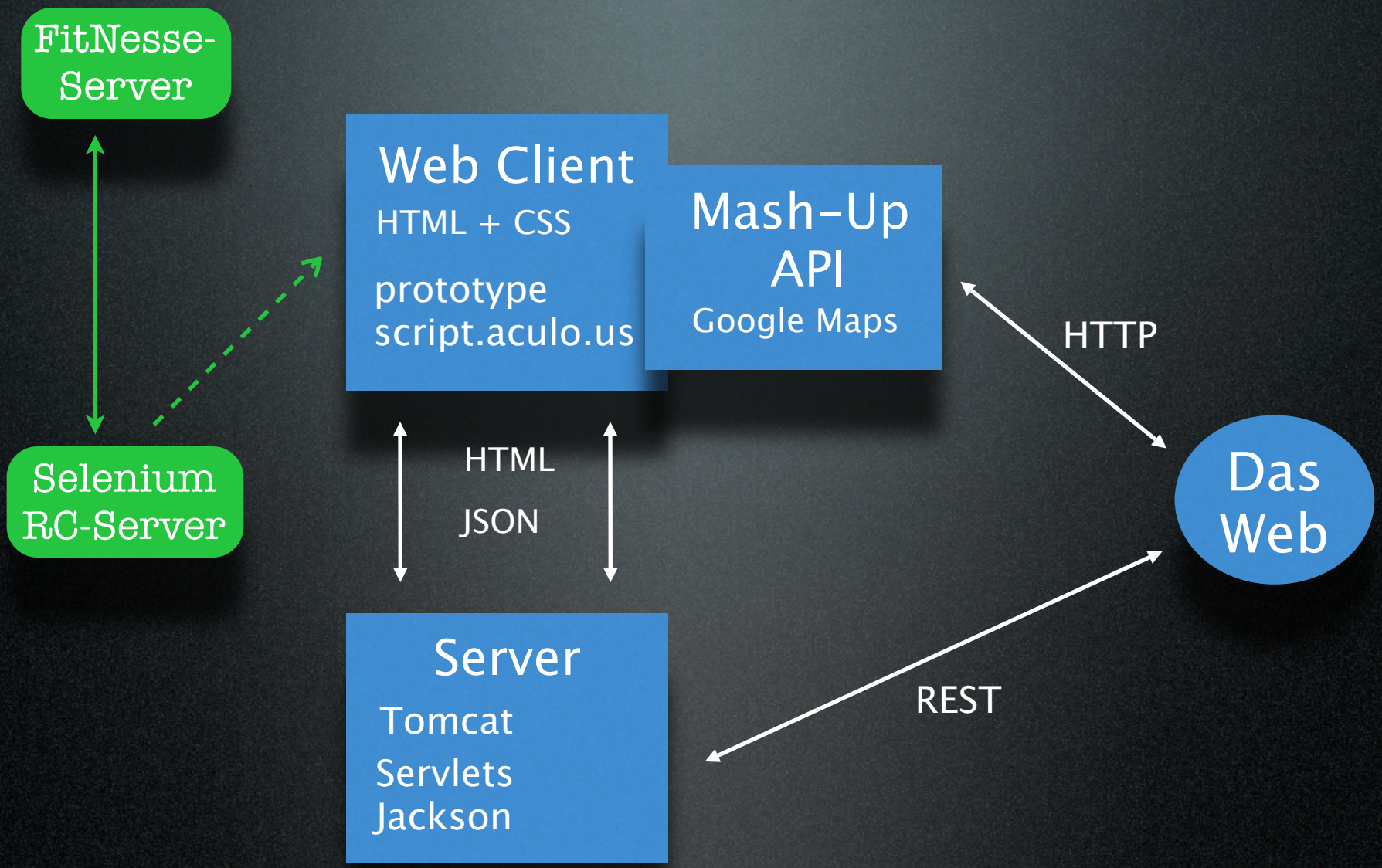

Web Client
HTML + CSS
prototype
script.aculo.us
Mash-Up
API
Google Maps
HTTP
HTML
JSON
Das
Web
Server
Tomcat
Servlets
Jackson
REST

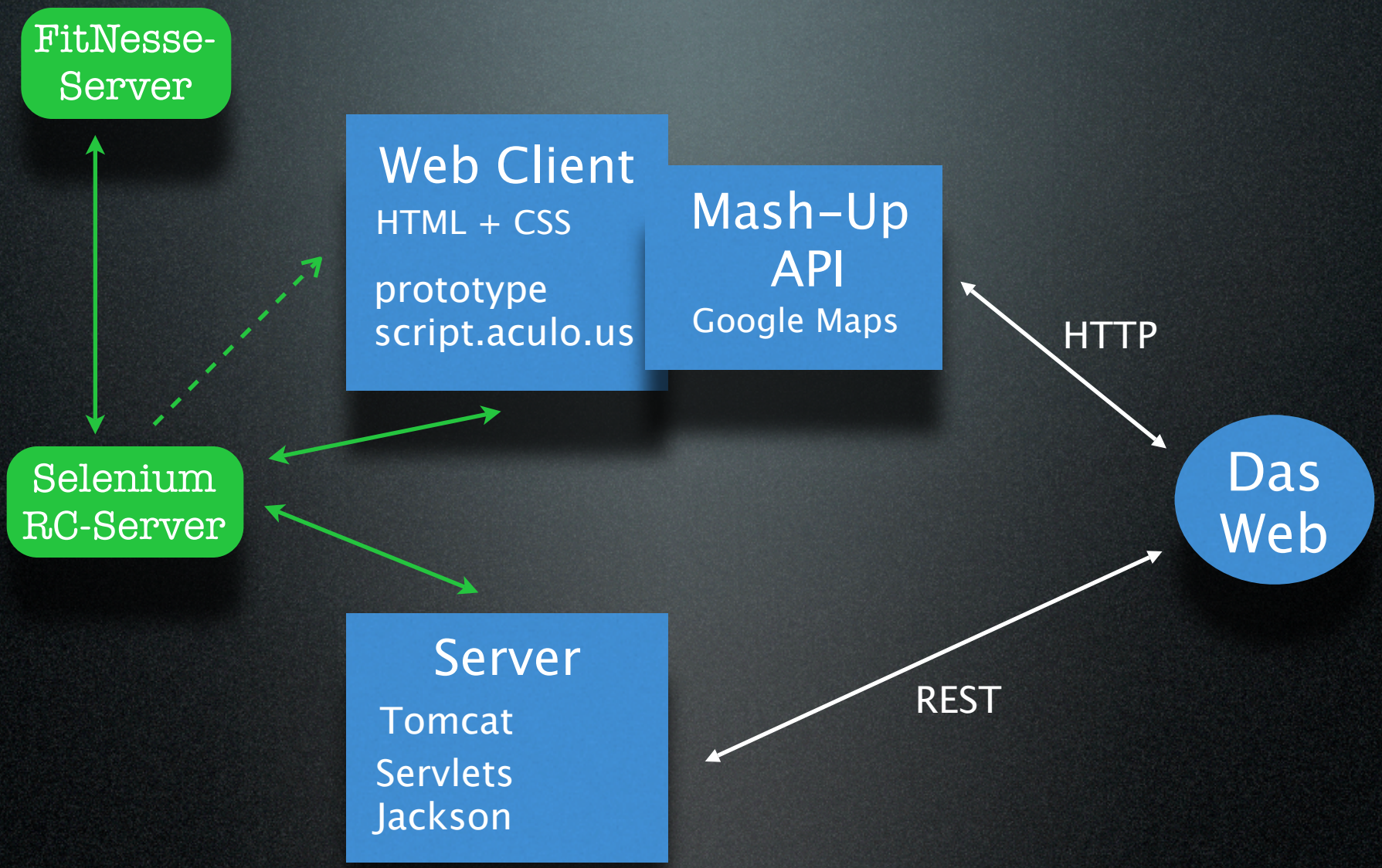
FitNesse-Server

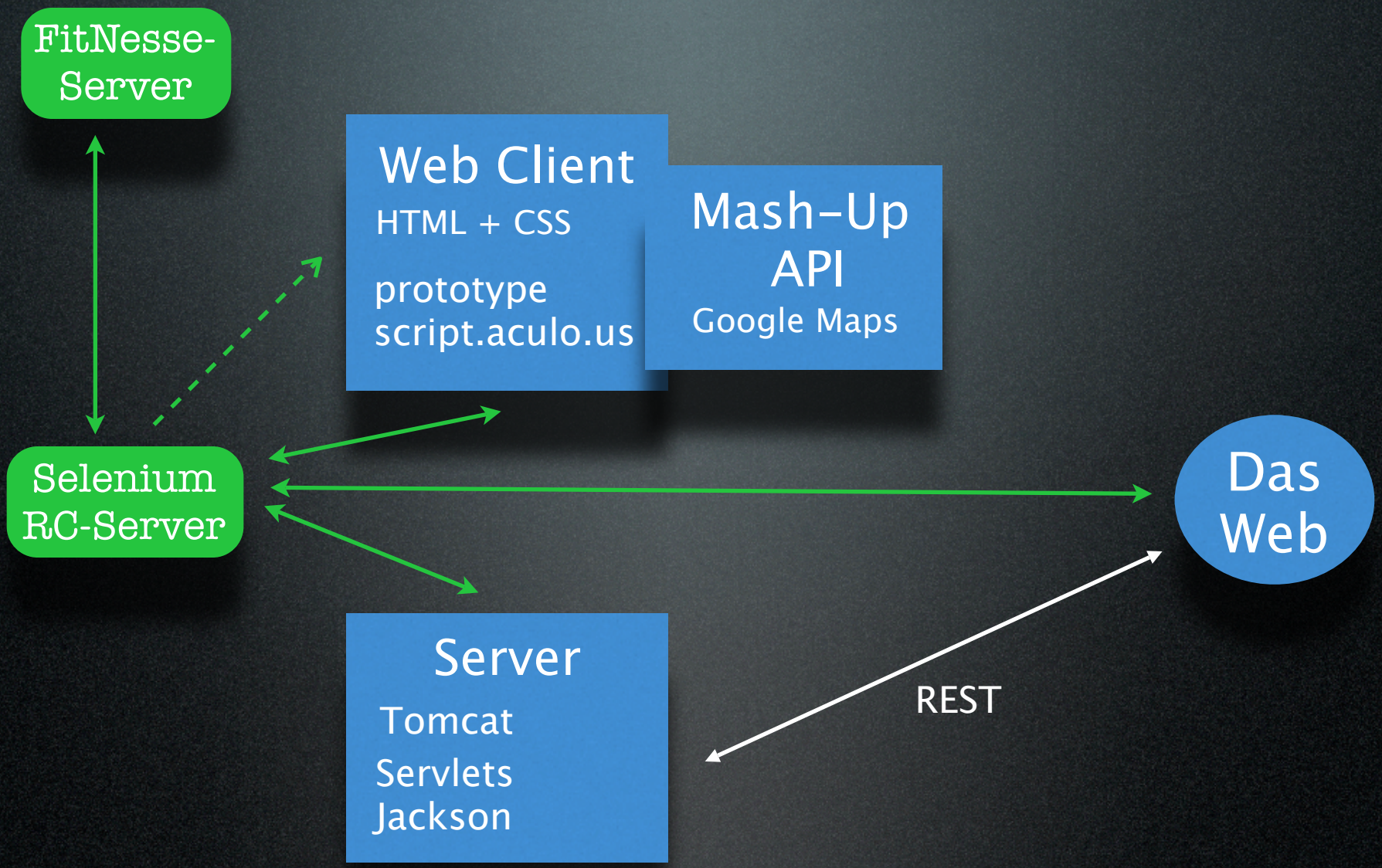



FitNesse-Server
Selenium RC-Server
Web Client
HTML + CSS
prototype
script.aculo.us
Mash-Up API
Google Maps
HTTP
HTML
JSON
Das Web
Server
Tomcat
Servlets
Jackson
REST


FitNesse-Server
Selenium RC-Server
Web Client
HTML + CSS
prototype
script.aculo.us
Mash-Up API
Google Maps
HTTP
HTML
JSON
Das Web
Server
Tomcat
Servlets
Jackson
REST


FitNesse-Server
Selenium RC-Server
Web Client
HTML + CSS
prototype
script.aculo.us
Mash-Up API
Google Maps
HTTP
Das Web
HTML
JSON
Server
Tomcat
Servlets
Jackson
REST


FitNesse-Server
Web Client
HTML + CSS
prototype
script.aculo.us
Mash-Up API
Google Maps
HTTP
Selenium RC-Server
Das Web
Server
Tomcat
Servlets
Jackson
REST


FitNesse-Server
Web Client
HTML + CSS
prototype
script.aculo.us
Mash-Up API
Google Maps
Selenium RC-Server
Das Web
Server
Tomcat
Servlets
Jackson
REST

Add participant from Totonto

start fresh

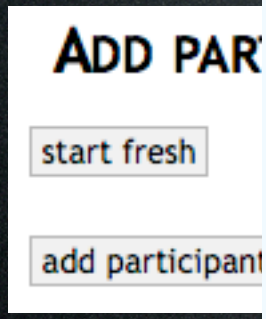
add participant from Toronto

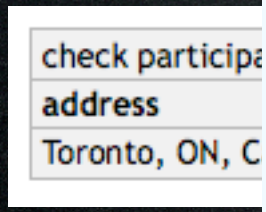
Add participant from Totonto

start fresh

add participant from	Toronto

check participants addresses	
address	**count**
Toronto, ON, Canada	1






```
public class SeleniumELOFacade implements ELOFacade...

    public List<ParticipantAddress> getParticipantsAddresses() {
        changeScreen(MENU_MY_EVENT);
        List<ParticipantAddress> addresses =
          new ArrayList<ParticipantAddress>();
        SeleniumXPathIterator i =
          new SeleniumXPathIterator(selenium,
            "//ul[@id='participantsAddresses']/li");
        for (String eachXPath : i) {
            String address =
              selenium.getText("xpath=" + eachXPath + "/span[1]");
            int count = getCount(eachXPath);
            addresses.add(new ParticipantAddress(address, count));
        }
        return addresses;
    }
```

Add participant from Totonto

start fresh

add participant from	Toronto

check participants addresses	
address	**count**
Toronto, ON, Canada	1

Add participant from Totonto

start fresh

add participant from	Toronto

check participants addresses	
address	**count**
Toronto, ON, Canada	1

map		
check	focus lattitude	43.8
check	focus longitude	-79.1
check	zoom	10
has event icon		
markers		
longitude	lattitude	

```
public class SeleniumELOFacade implements ELOFacade...

    public EventPlanningMap map() {
        changeScreen(MENU_MY_EVENT);
        SeleniumTableRetriever tableRetriever =
          new SeleniumTableRetriever(selenium, TABLE_MAP_VALUES);
        Map<String, List<String>> mapValues =
          tableRetriever.asMap();
        MapLocation focus = ...
        int zoom = ...
        EventPlanningMap gmap = new EventPlanningMap(focus, zoom);
        eventPlanningMap.setHasEventIcon(hasEventIcon);
        addMarkersToMap(eventPlanningMap, mapValues);
        return gmap;
    }
```

ADD PAR

start fresh

add participan

check participa
address
Toronto, ON, C

map		
check	focus lattitude	43.8
check	focus longitude	-79.1
check	zoom	10
has event icon		
markers		
longitude	lattitude	

Add participant from Totonto

start fresh

add participant from	Toronto

check participants addresses	
address	**count**
Toronto, ON, Canada	1

map		
check	focus lattitude	43.8
check	focus longitude	-79.1
check	zoom	10
has event icon		
markers		
longitude	lattitude	

set up geocoder			
query	longitude	lattitude	address
berlin	13.41	52.52	Berlin, Germany
toronto	-79.1	43.8	Toronto, ON, Canada

Add participant from Totonto

start fresh

add participant from	Toronto

check participants addresses	
address	**count**
Toronto, ON, Canada	1

map		
check	focus lattitude	43.8
check	focus longitude	-79.1
check	zoom	10
has event icon		
markers		
longitude	lattitude	

set up geocoder			
query	longitude	lattitude	address
berlin	13.41	52.52	Berlin, Germany
toronto	-79.1	43.8	Toronto, ON, Canada

```
public class SetUpGeocoder extends SetUpFixture...
    private final WebSimulatorBackdoor backdoor;
    public void queryLongitudeLattitudeAddress(String query,
      String longitude, String lattitude, String address) {
        backdoor.addFakeGeoCoding(query, lattitude, longitude, address);
    }
```

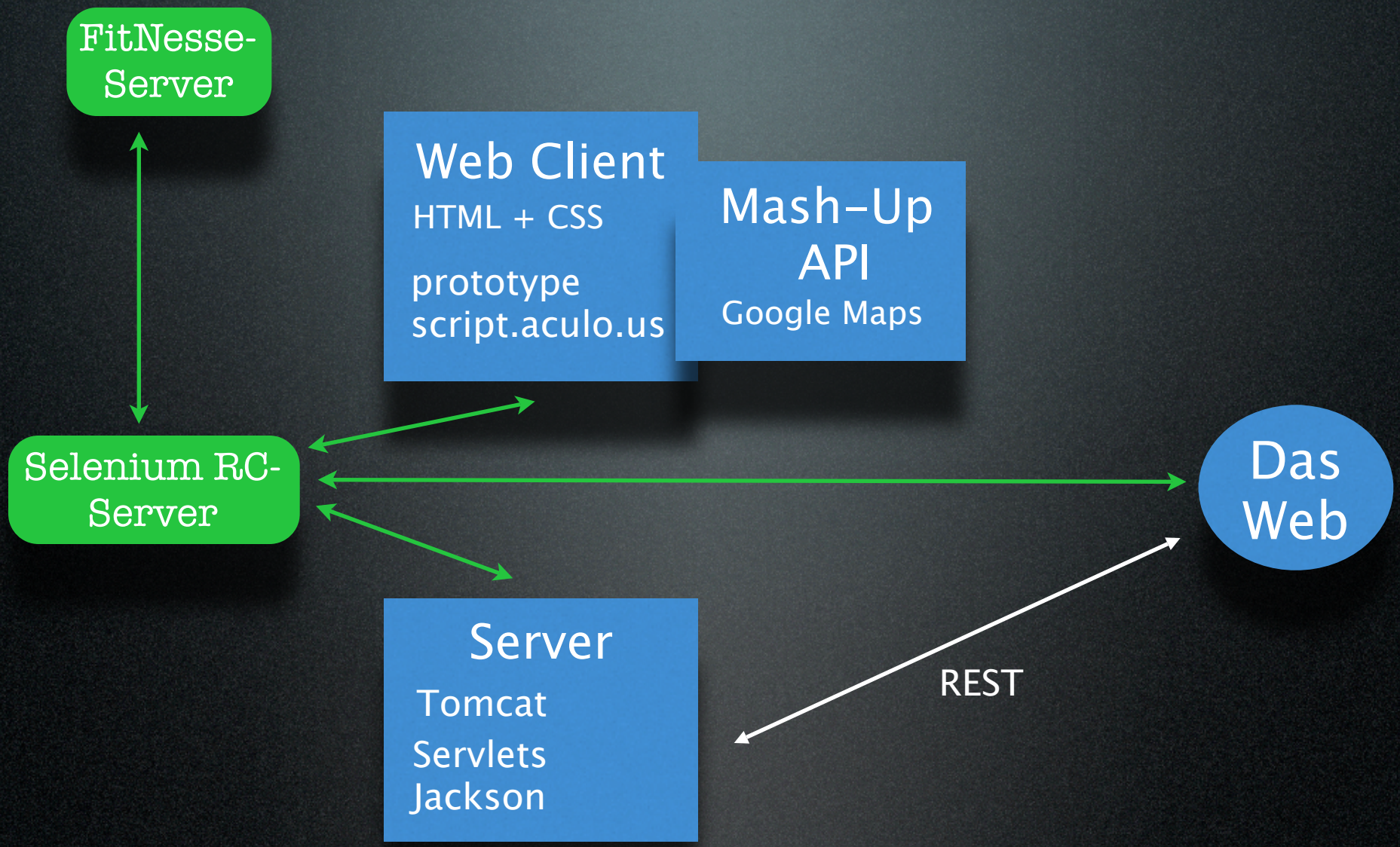
check participants addresses	
address	**count**
Toronto, ON, Canada	1

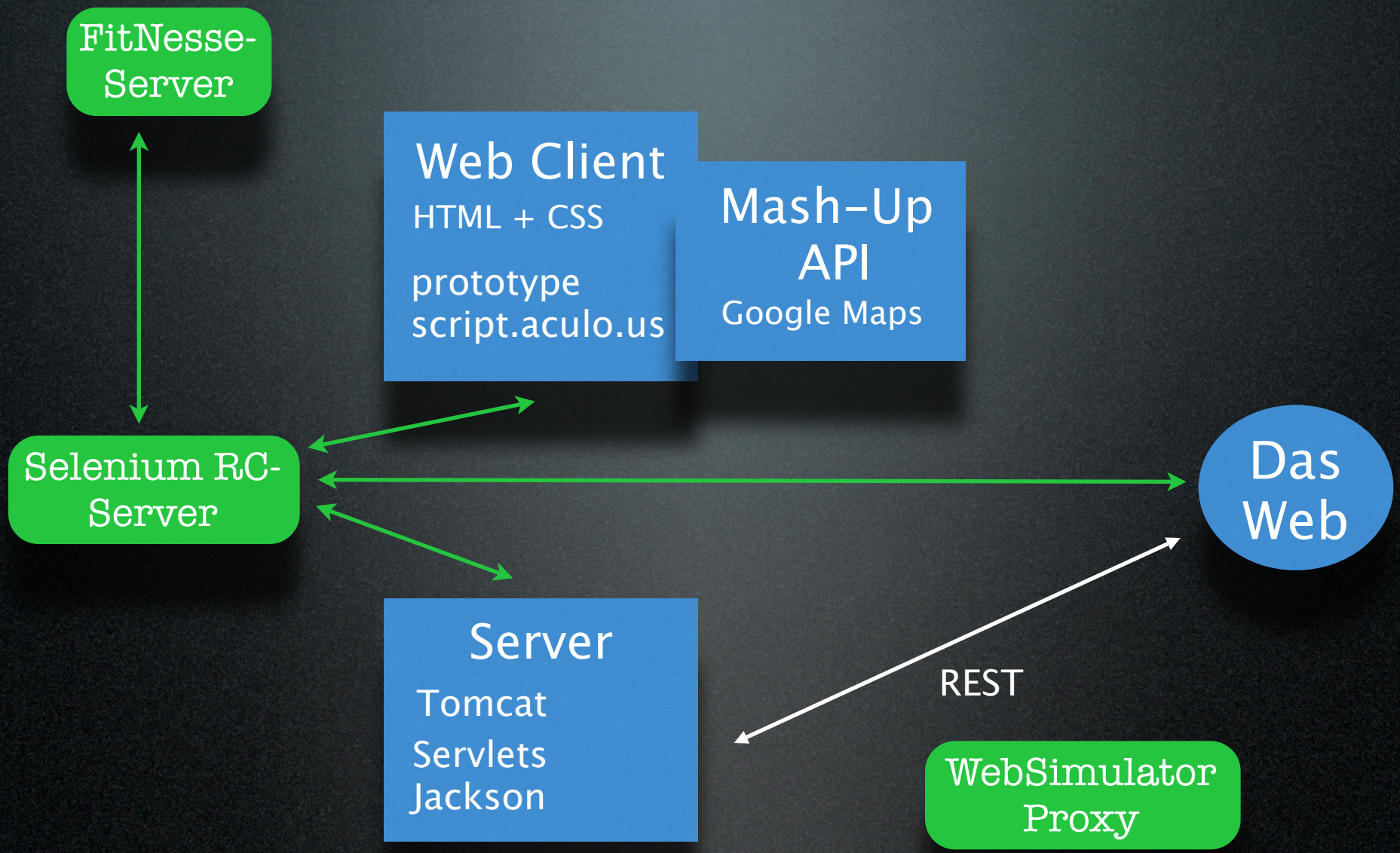
map		
check	focus lattitude	43.8
check	focus longitude	-79.1
check	zoom	10
has event icon		
markers		
longitude	lattitude	

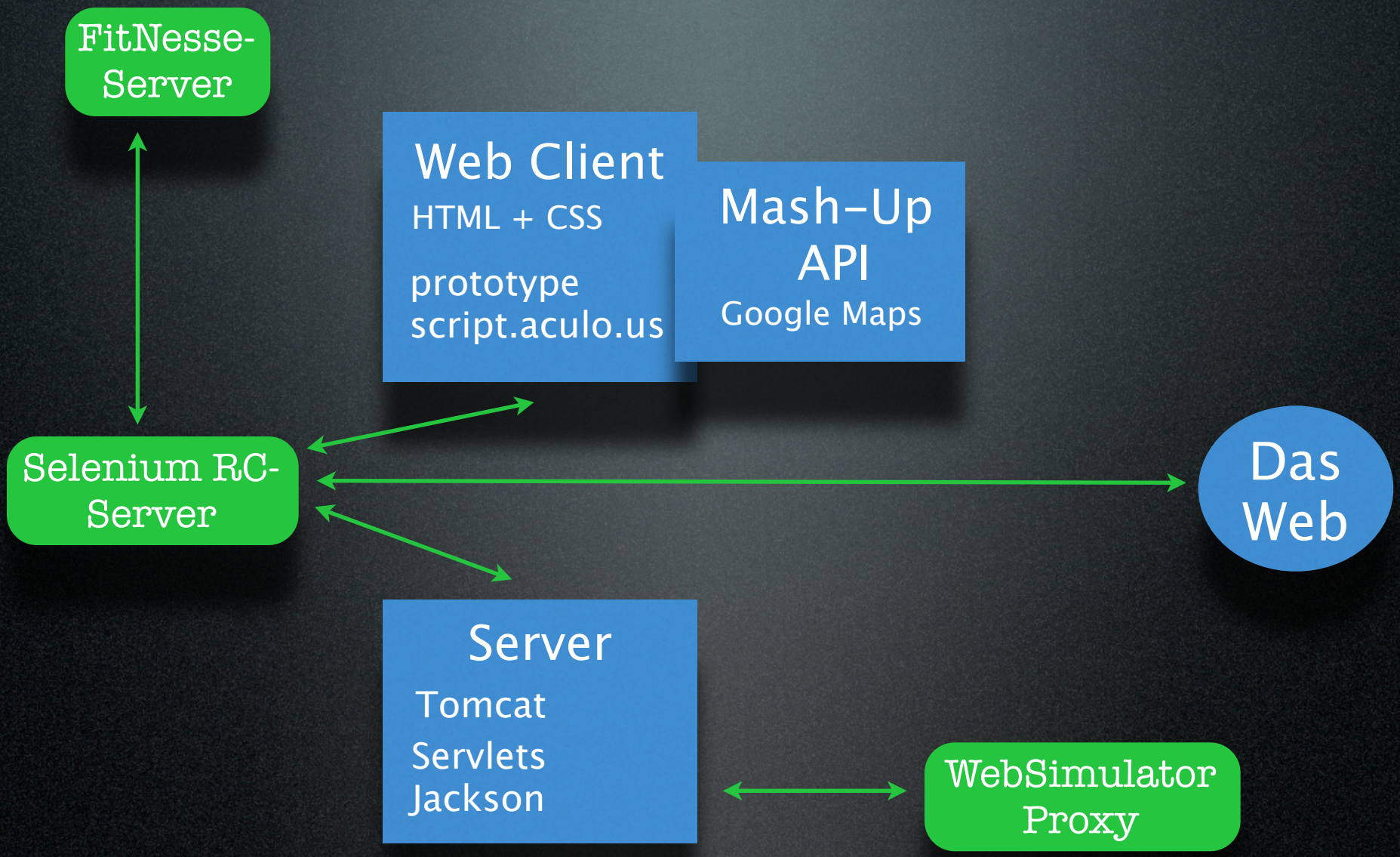
set up geocoder			
query	longitude	lattitude	address
berlin	13.41	52.52	Berlin, Germany
toronto	-79.1	43.8	Toronto, ON, Canada

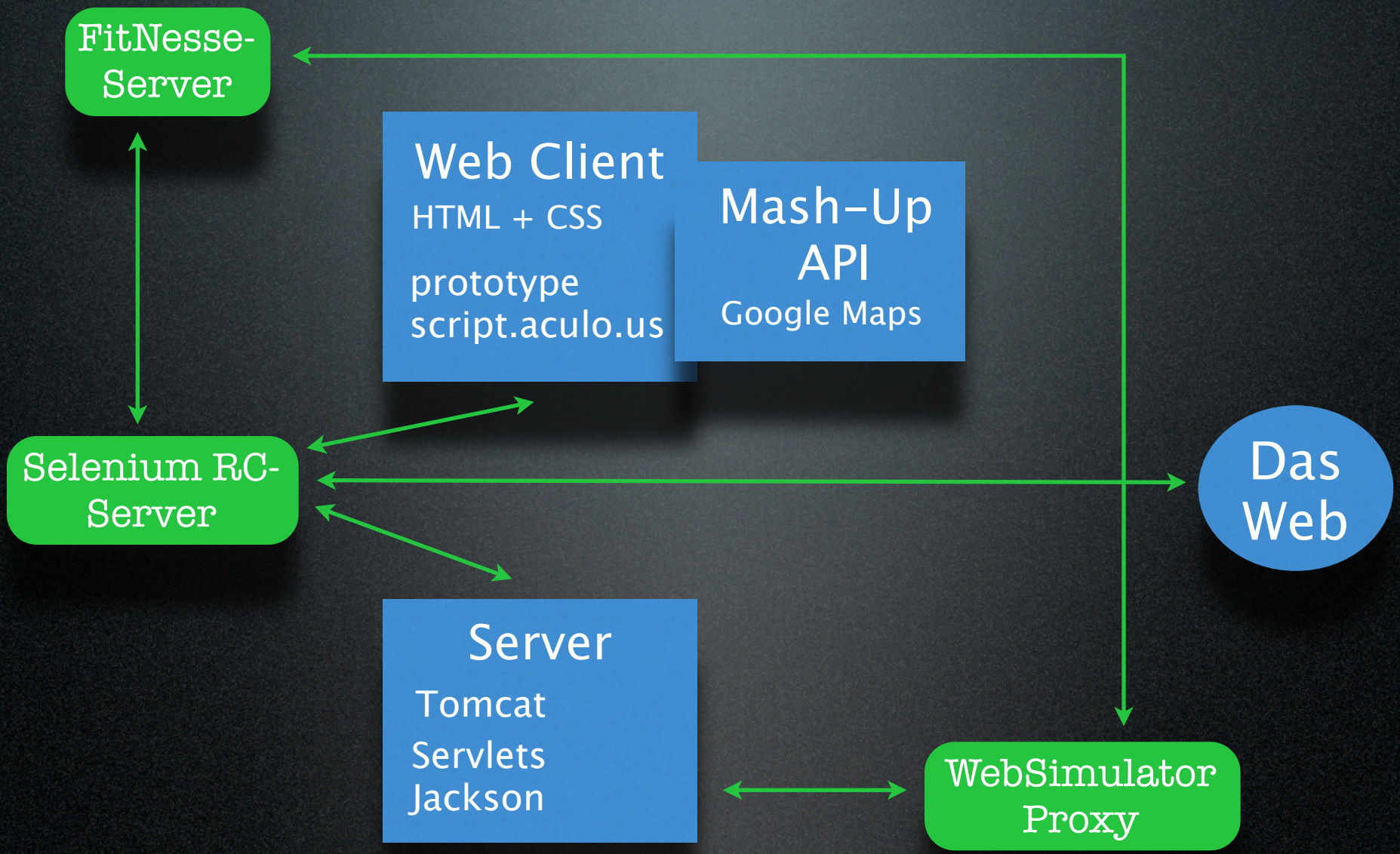
```
public class SetUpGeocoder extends SetUpFixture...
    private final WebSimulatorBackdoor backdoor;
    public void queryLongitudeLattitudeAddress(String query,
      String longitude, String lattitude, String address) {
        backdoor.addFakeGeoCoding(query, lattitude, longitude, address);
    }
```

```
public class WebSimulatorBackdoor extends WebBackdoor...

    public void addFakeGeoCoding(String query, String lattitude,
      String longitude, String address) {
        Map<String, String> params = new HashMap<String, String>();
        params.put("query", query);
        params.put("address", address);
        params.put("longitude", longitude);
        params.put("lattitude", lattitude);
        executeRequest("addFakeGeoCoding", params);
    }
```


FitNesse-Server
Web Client
HTML + CSS
prototype
script.aculo.us
Mash-Up API
Google Maps
Selenium RC-Server
Das Web
Server
Tomcat
Servlets
Jackson
REST


FitNesse-Server
Web Client
HTML + CSS
prototype
script.aculo.us
Mash-Up API
Google Maps
Selenium RC-Server
Das Web
Server
Tomcat
Servlets
Jackson
REST
WebSimulator Proxy

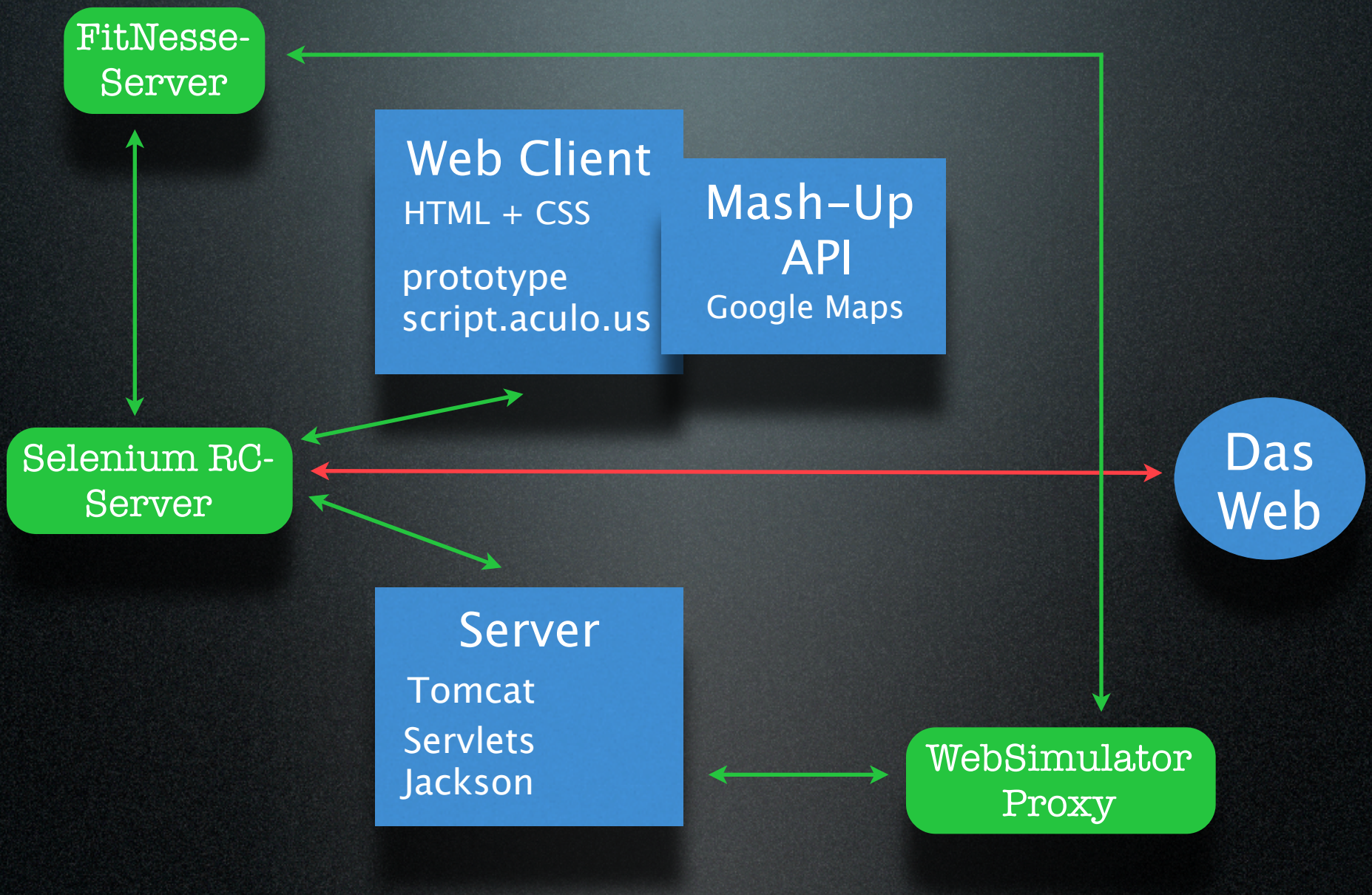

FitNesse-Server
Web Client
HTML + CSS
prototype
script.aculo.us
Mash-Up API
Google Maps
Selenium RC-Server
Das Web
Server
Tomcat
Servlets
Jackson
WebSimulator Proxy

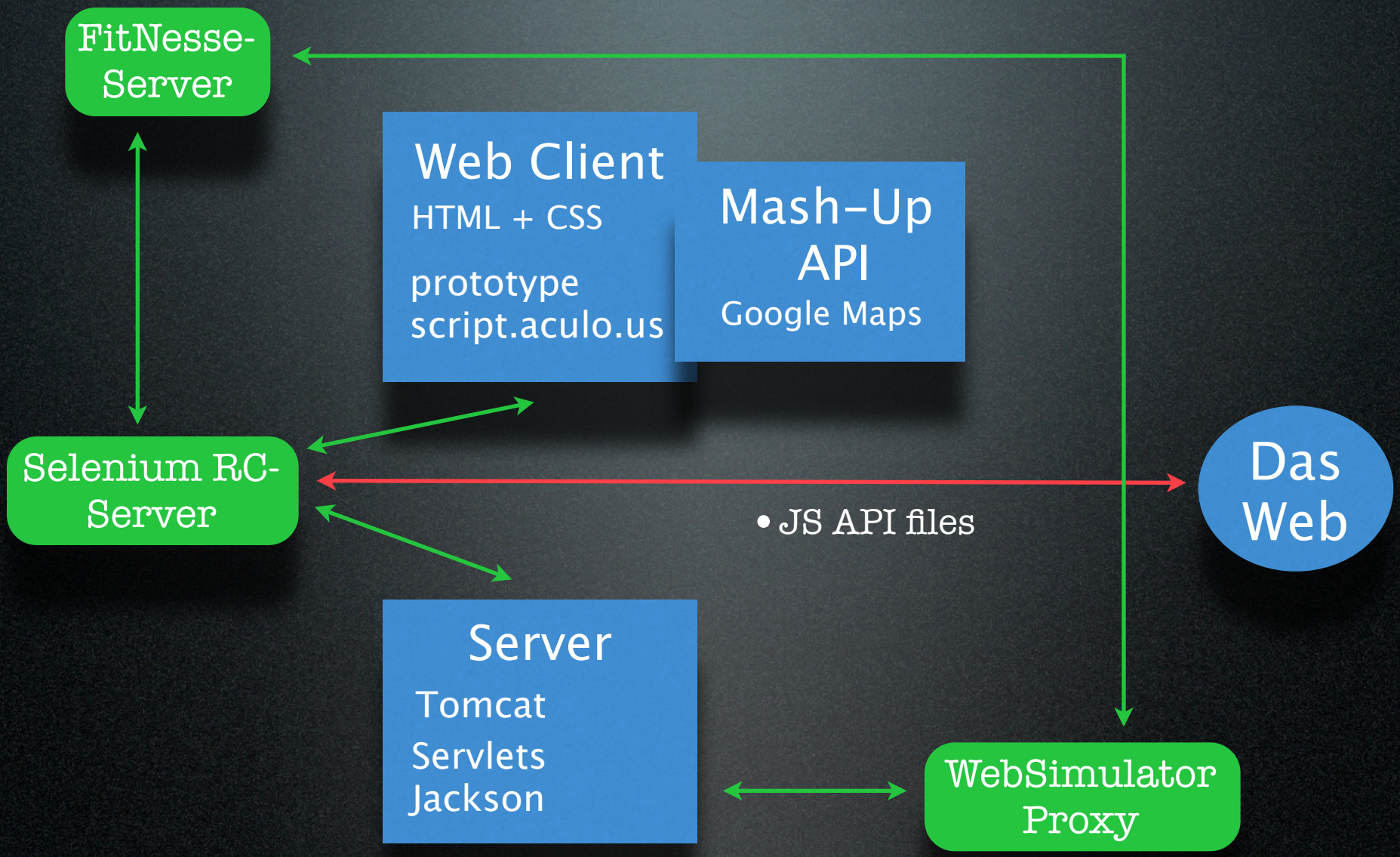

FitNesse-Server
Web Client
HTML + CSS
prototype
script.aculo.us
Mash-Up API
Google Maps
Selenium RC-Server
Das Web
Server
Tomcat
Servlets
Jackson
WebSimulator Proxy

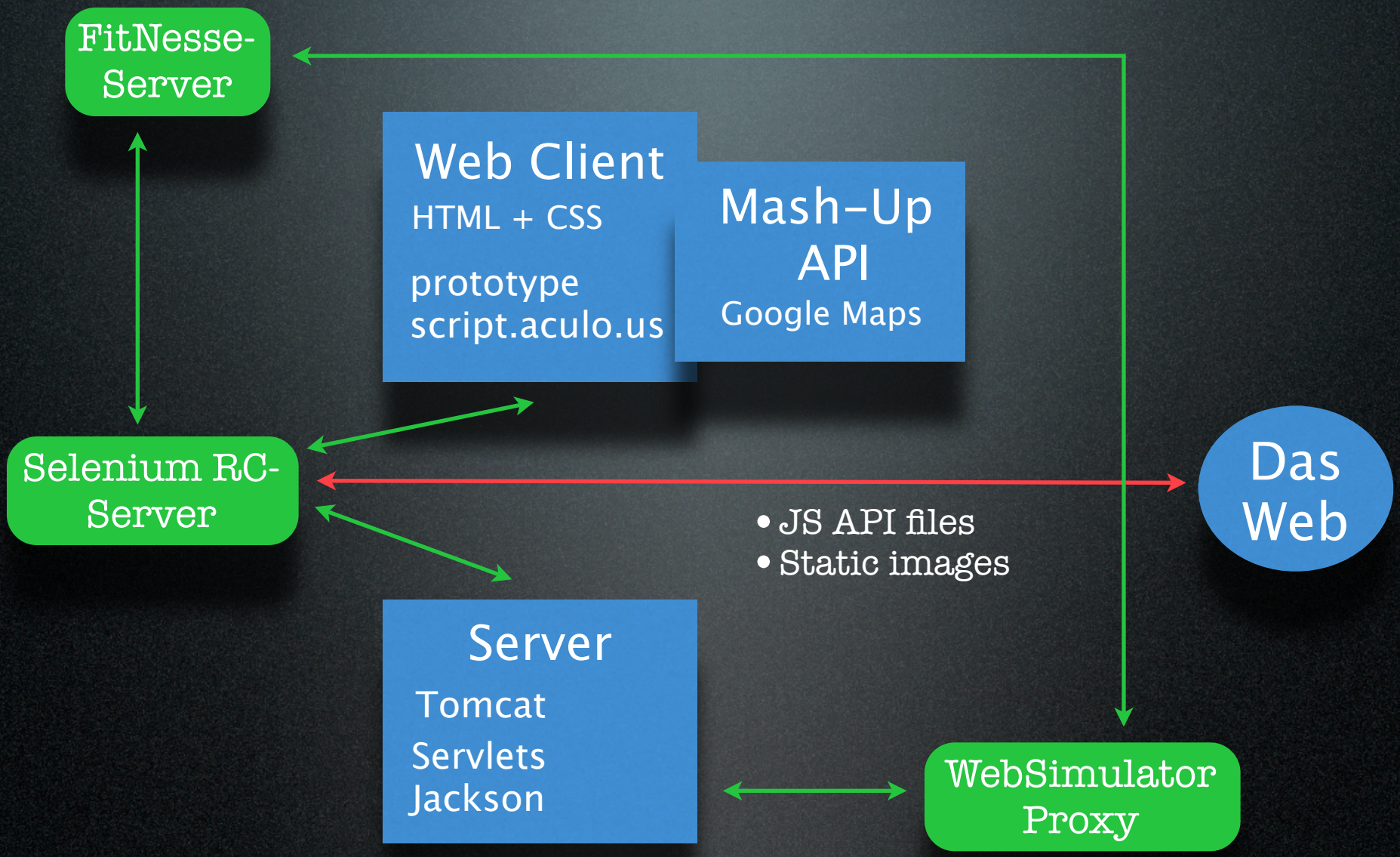
Sind wir jetzt Web-unabhängig?

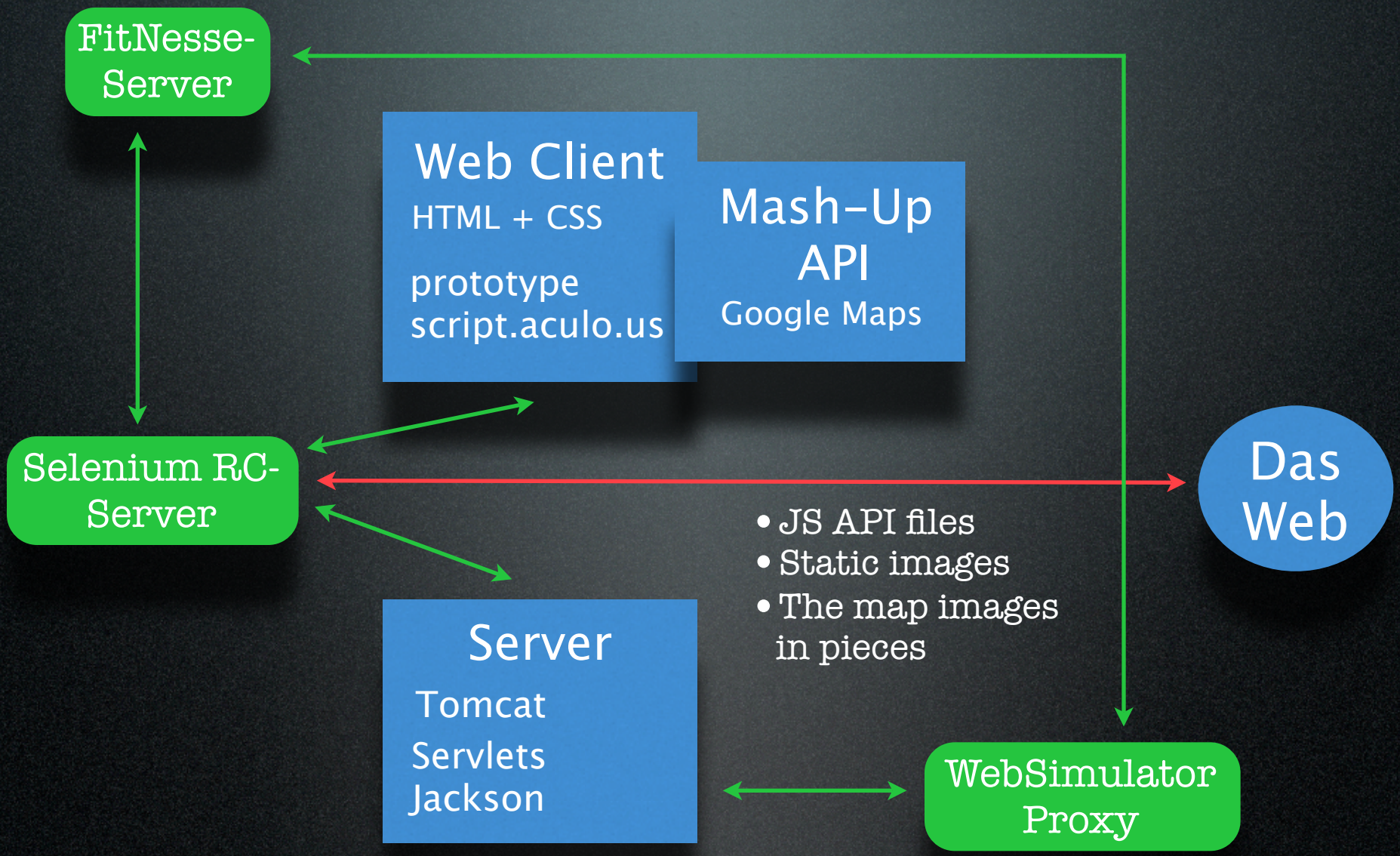
Sind wir jetzt Web-unabhängig?

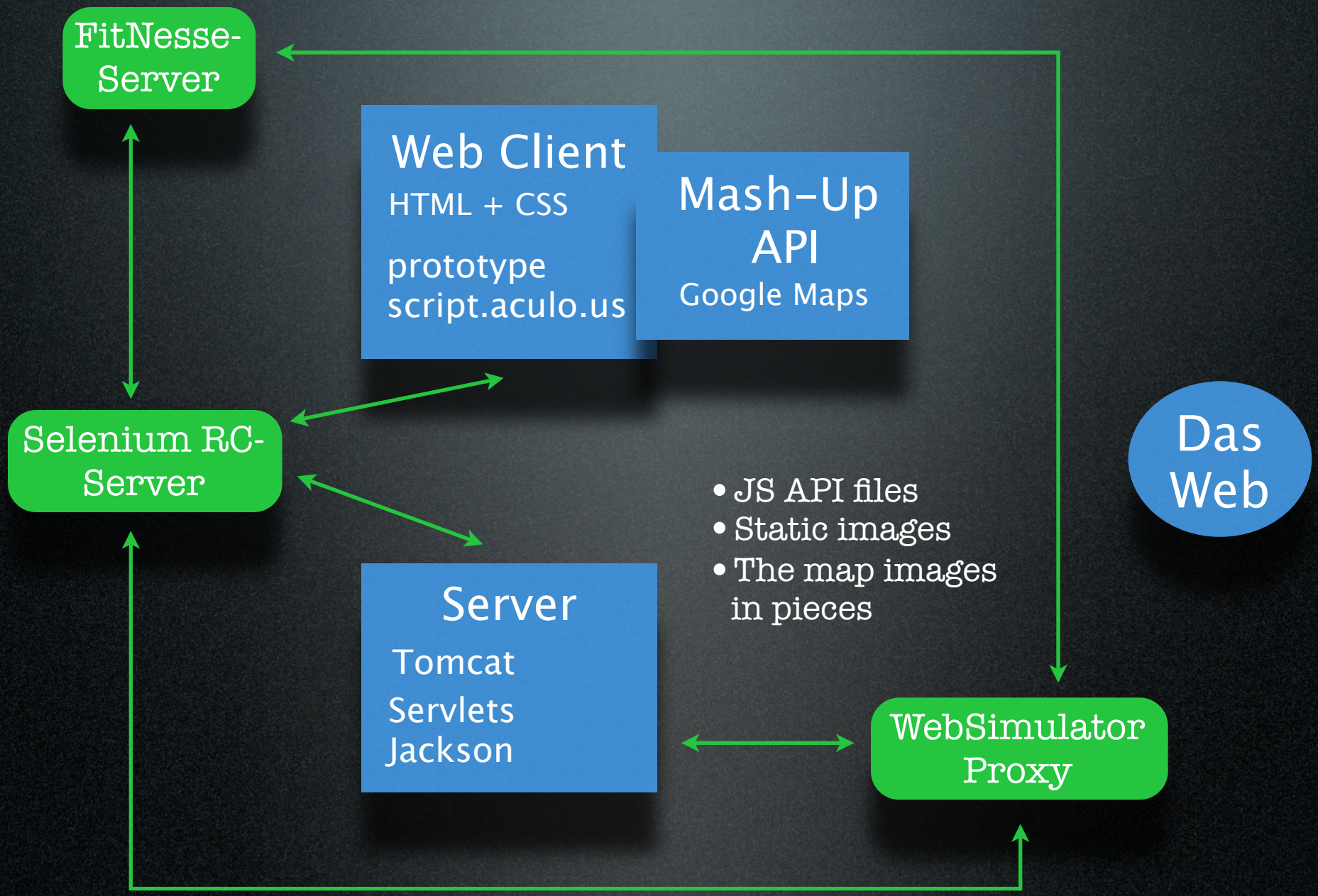
Noch nicht!

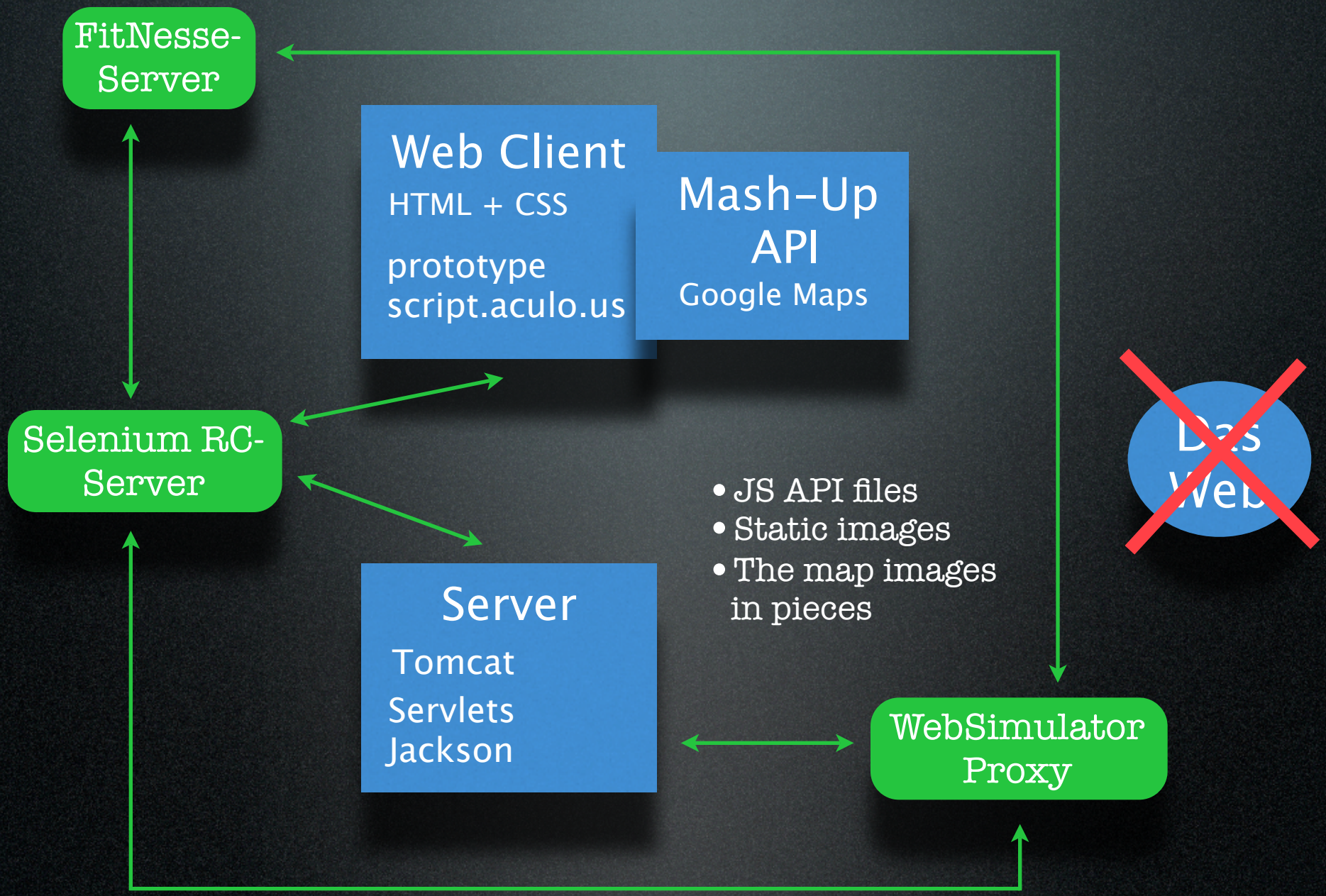

FitNesse-Server
Web Client
HTML + CSS
prototype
script.aculo.us
Mash-Up API
Google Maps
Selenium RC-Server
Das Web
Server
Tomcat
Servlets
Jackson
WebSimulator Proxy


FitNesse-Server
Web Client
HTML + CSS
prototype
script.aculo.us
Mash-Up API
Google Maps
Selenium RC-Server
• JS API files
Das Web
Server
Tomcat
Servlets
Jackson
WebSimulator Proxy


FitNesse-Server
Web Client
HTML + CSS
prototype
script.aculo.us
Mash-Up API
Google Maps
Selenium RC-Server
Das Web
• JS API files
• Static images
Server
Tomcat
Servlets
Jackson
WebSimulator Proxy


FitNesse-Server
Web Client
HTML + CSS
prototype
script.aculo.us
Mash-Up API
Google Maps
Selenium RC-Server
Das Web
• JS API files
• Static images
• The map images in pieces
Server
Tomcat
Servlets
Jackson
WebSimulator Proxy


FitNesse-Server
Web Client
HTML + CSS
prototype
script.aculo.us
Mash-Up API
Google Maps
Selenium RC-Server
Das Web
• JS API files
• Static images
• The map images in pieces
Server
Tomcat
Servlets
Jackson
WebSimulator Proxy


FitNesse-Server
Web Client
HTML + CSS
prototype
script.aculo.us
Mash-Up API
Google Maps
Selenium RC-Server
Das Web
• JS API files
• Static images
• The map images in pieces
Server
Tomcat
Servlets
Jackson
WebSimulator Proxy

Die Tests über die Business-Facade zu fahren...

... ist viel weniger Umstand!

... ist stabiler!

Die Tests über die Business-Facade zu fahren...

... ist viel weniger Umstand!

... ist stabiler!

Man benötigt...

Die Tests über die Business-Facade zu fahren...

... ist viel weniger Umstand!

... ist stabiler!

Man benötigt...

- **keinen** Servlet-Container für das Deployment

Die Tests über die Business-Facade zu fahren...

... ist viel weniger Umstand!

... ist stabiler!

Man benötigt...

- **keinen** Servlet-Container für das Deployment
- **keinen** Web-Browser

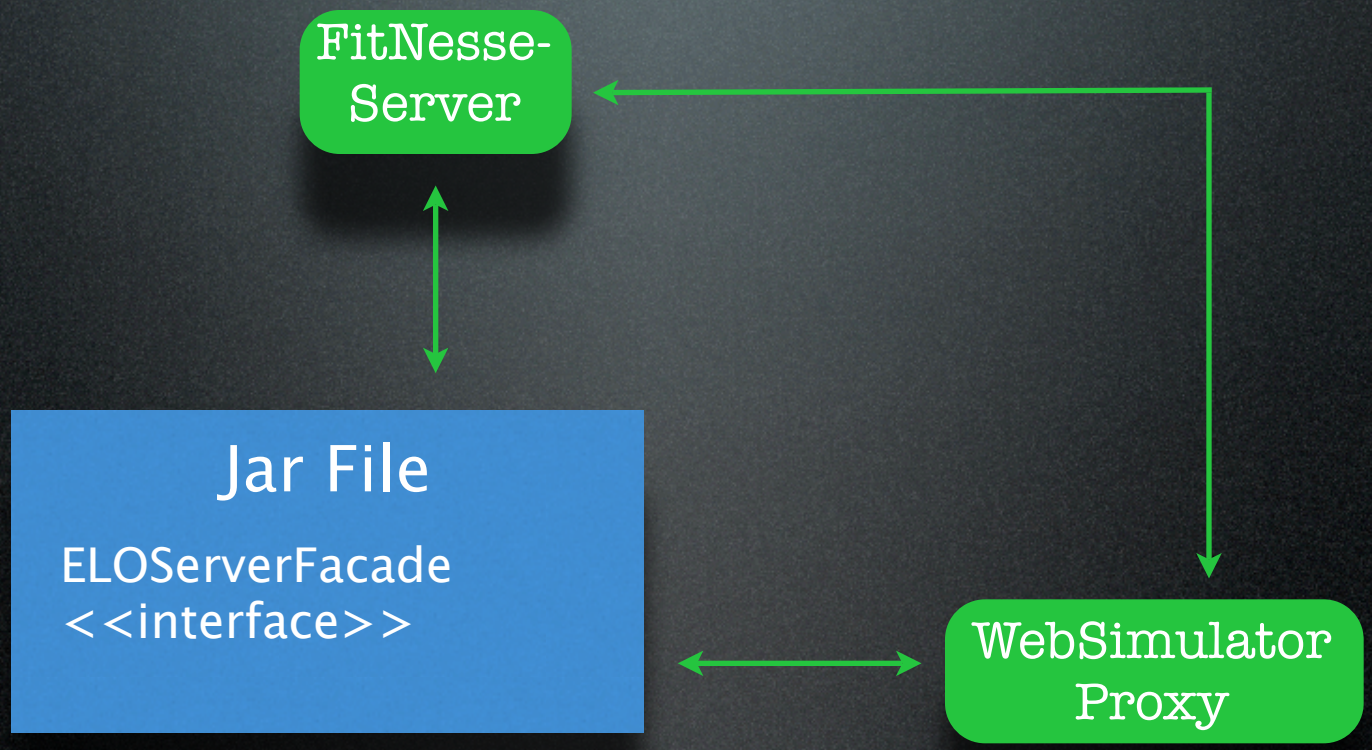
Die Tests über die Business-Facade zu fahren...

... ist viel weniger Umstand!

... ist stabiler!

Man benötigt...

- **keinen** Servlet-Container für das Deployment
- **keinen** Web-Browser
- **kein** Selenium RC


FitNesse-
Server
Jar File
ELOServerFacade
<<interface>>
WebSimulator
Proxy

Laufzeiten der Fitnesse-Tests

Laufzeiten der Fitnesse-Tests

- Mit Selenium-RC: 108 sec

Laufzeiten der Fitnesse-Tests

- Mit Selenium-RC: 108 sec
- Mit Business-Fassade: 8 sec

Laufzeiten der Fitnesse-Tests

- Mit Selenium-RC: 108 sec
- Mit Business-Fassade: 8 sec
- (Mit HtmlUnit: 30 sec)

Heuristiken für Ajax Akzeptanztests

- Lass **wenige** Tests mit dem echten Browser laufen
  - Lass diese Tests aber auf allen Ziel-Browsern laufen
- Der Großteil der Tests sollte die Business-Fassade verwenden
- Strebe Web-Unabhängigkeit an
  - Aber übertreibe dabei nicht!

Testen der extternen Dienste

- Wie stabil ist die fremde API?
- Wie groß ist die Verfügbarkeit

Fazit

- Testgetriebenes Ajax ist machbar, aber
  - die Tests werden durch starke Asynchronität geprägt
  - das clientseitiges Toolset ist noch nicht zufriedenstellend
- Starke Verteilung im Web 2.0
  - viel Aufwand für Mocks und Simulationen
  - eventuell „Live Überwachung“ notwendig
- Teste mit allen Ziel-Browsern!

Web Resources

- http://mir.aculo.us/stuff/AdventuresInJavaScriptTesting.pdf
- http://ajaxian.com/by/topic/testing/
- http://ajaxpatterns.org/Browser-Side_Test
- http://blog.johanneslink.net/ajax-travelogue-part-3/

Questions?

http://www.slideshare.net/jlink/???/