

Testgetriebenes Ajax



Johannes Link
unabhängiger Softwarecoach
<http://johanneslink.net>

Marco Klemm
andrena objects ag
<http://andrena.de>

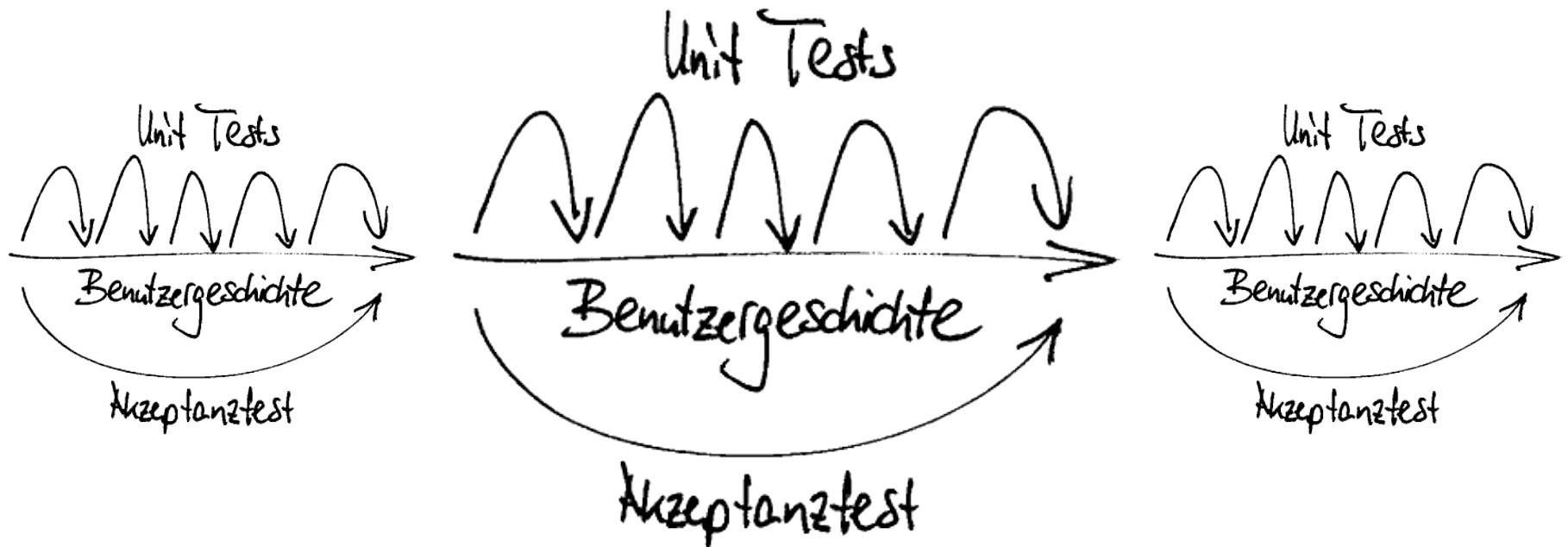
Agenda

- Was ist Testgetriebene Entwicklung (TDD)?
- Herausforderungen bei Ajax-Applikationen
- Techniken und Werkzeuge
- Beispielanwendung

Was ist testgetriebene Entwicklung?

- Testgetriebene Programmierung:
Motiviere jede Verhaltensänderung am Code durch einen automatisierten Test
- Refactoring:
Bringe den Code immer in die “einfache Form”
- Häufige Integration:
Integriere den Code so häufig wie nötig

Im Großen und im Kleinen...

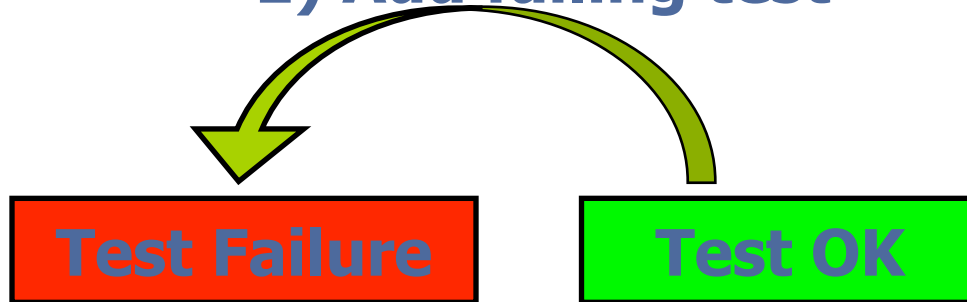


Test/Code/Refactor – Zyklus

Test OK

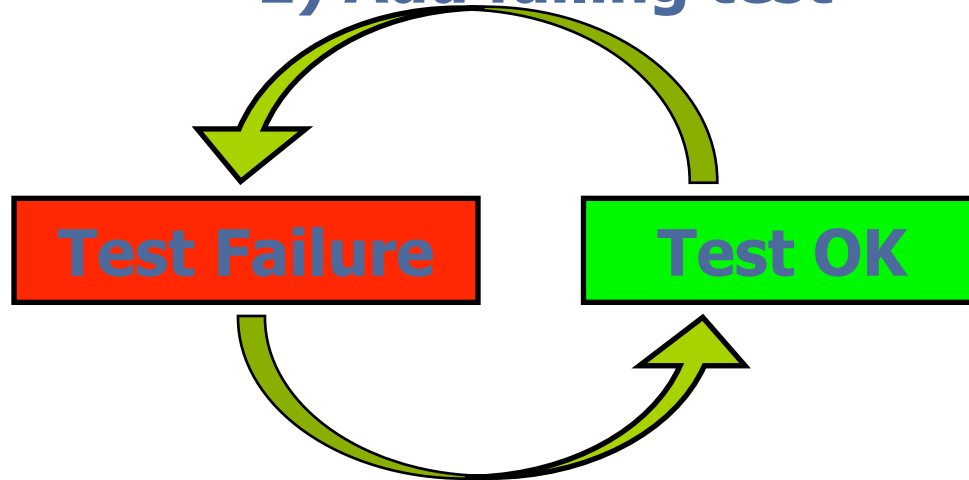
Test/Code/Refactor – Zyklus

1) Add failing test



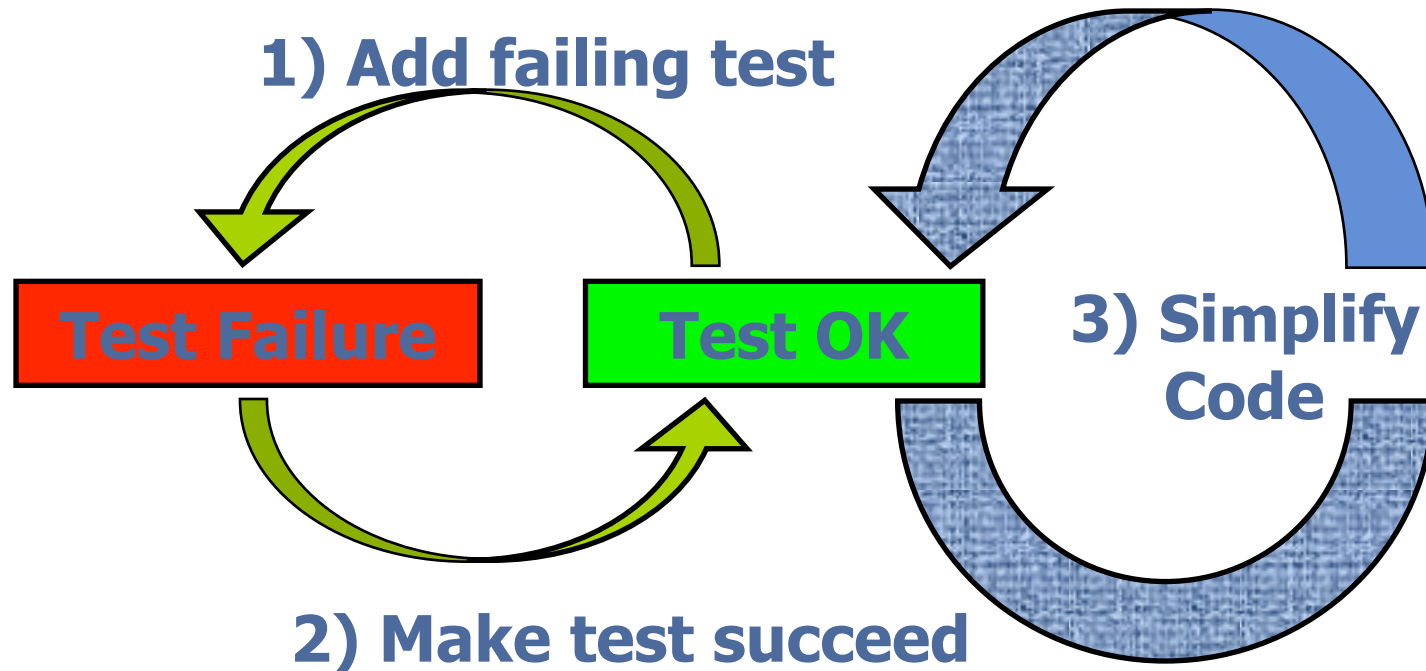
Test/Code/Refactor – Zyklus

1) Add failing test



2) Make test succeed

Test/Code/Refactor – Zyklus



Web-Client

HTML + CSS

JavaScript

Prototype

Scriptaculous

Dojo

DWR (client lib)

Web-Client

HTML + CSS
JavaScript
Prototype
Scriptaculous
Dojo
DWR (client lib)

Server

Tomcat, Apache
Java
Servlets
JSF
DWR (server lib)

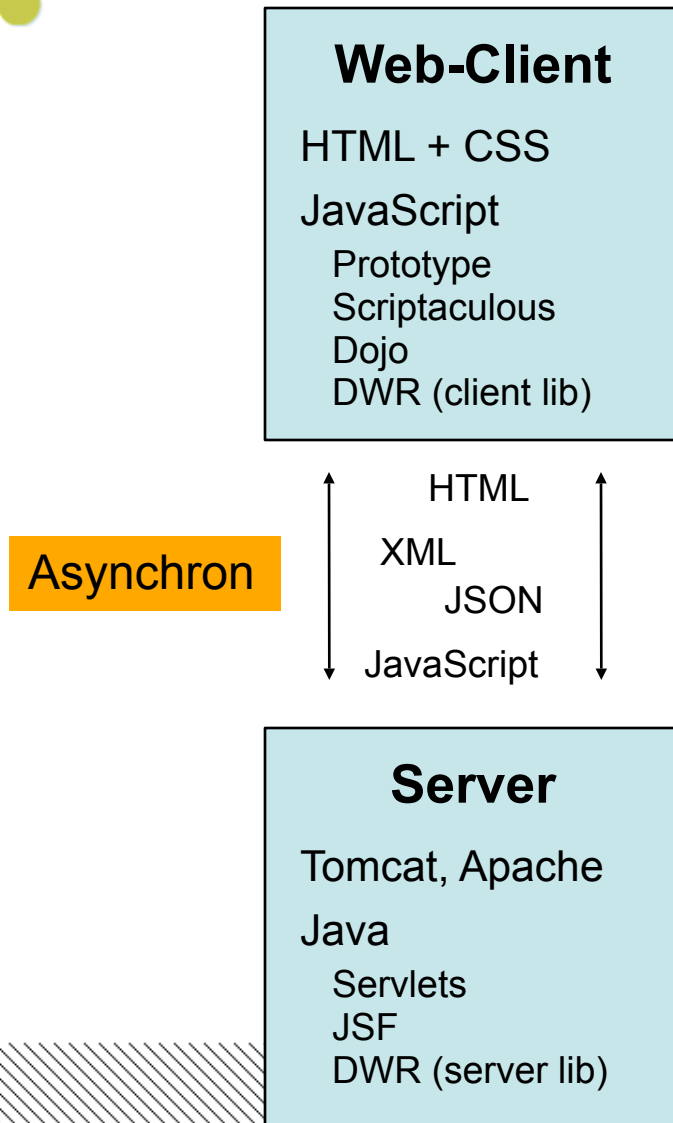
Web-Client

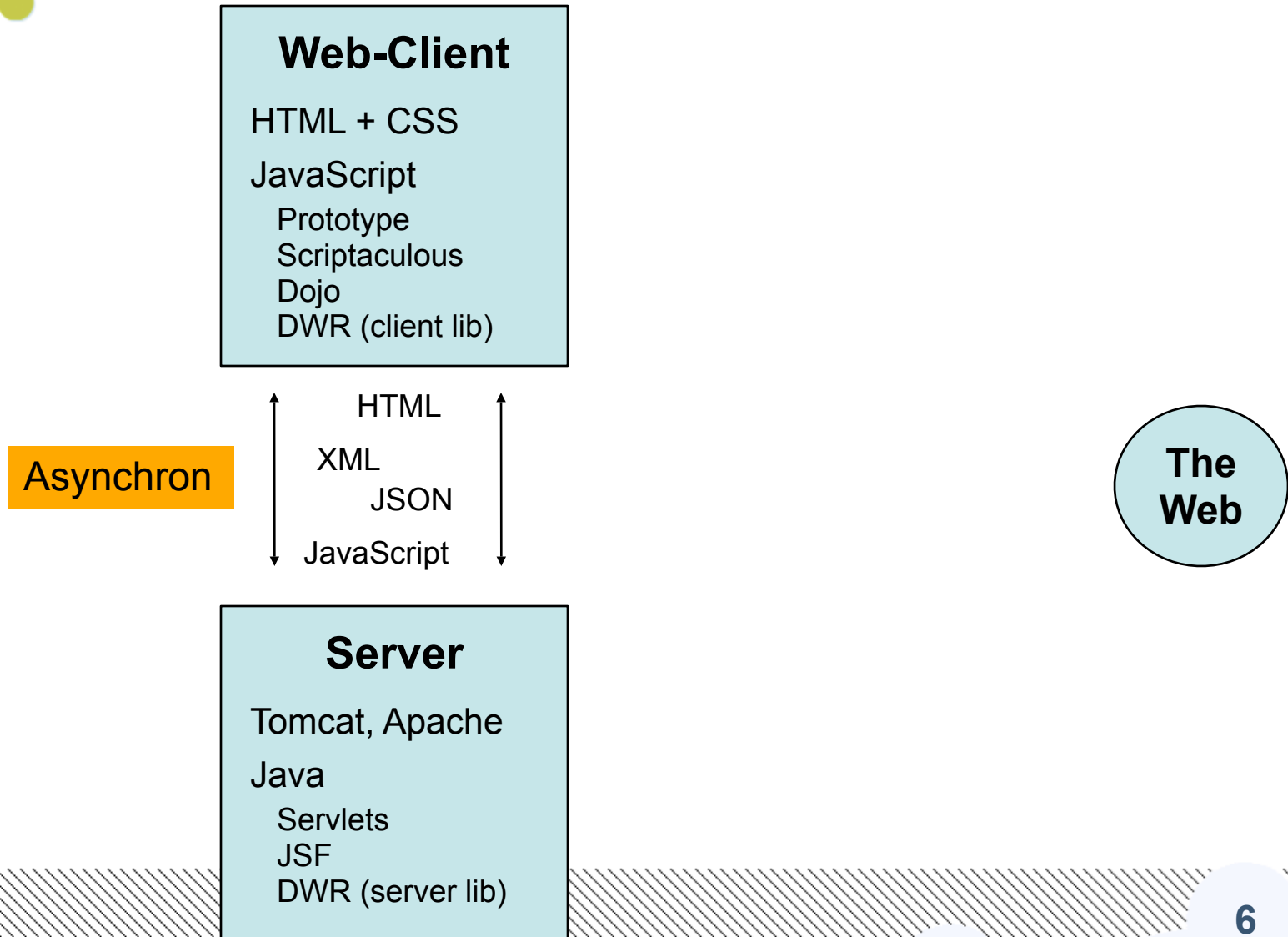
HTML + CSS
JavaScript
Prototype
Scriptaculous
Dojo
DWR (client lib)

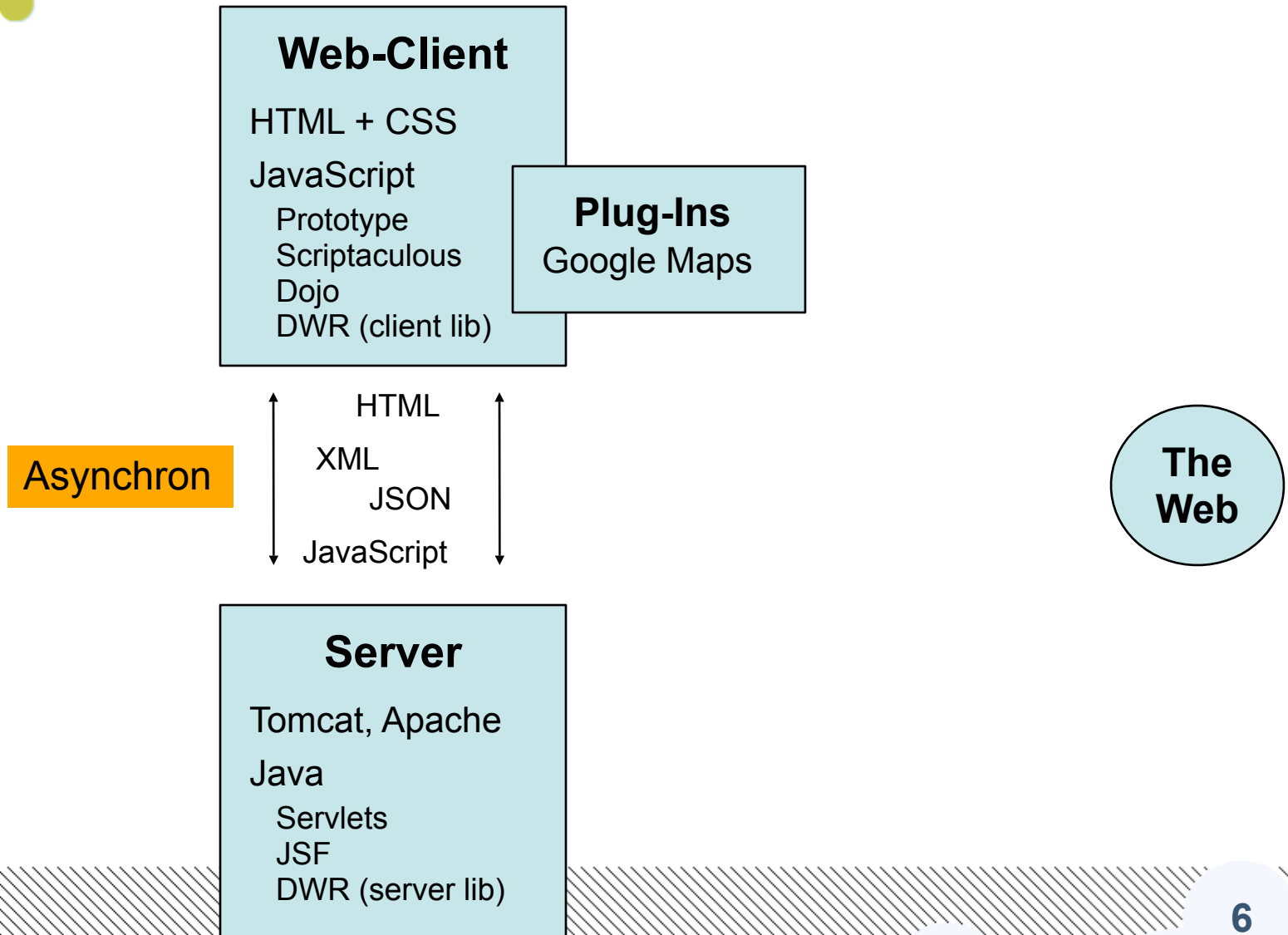


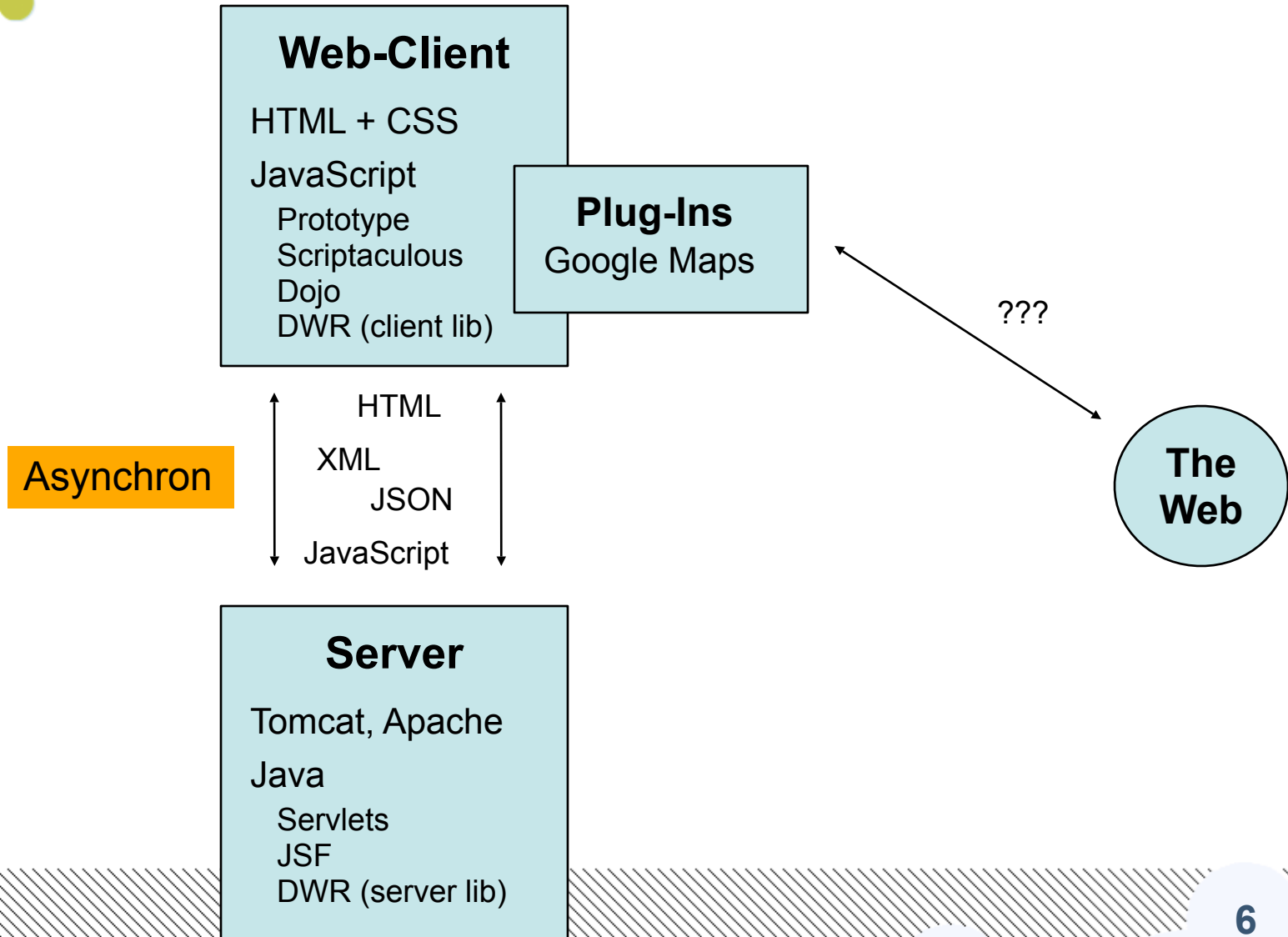
Server

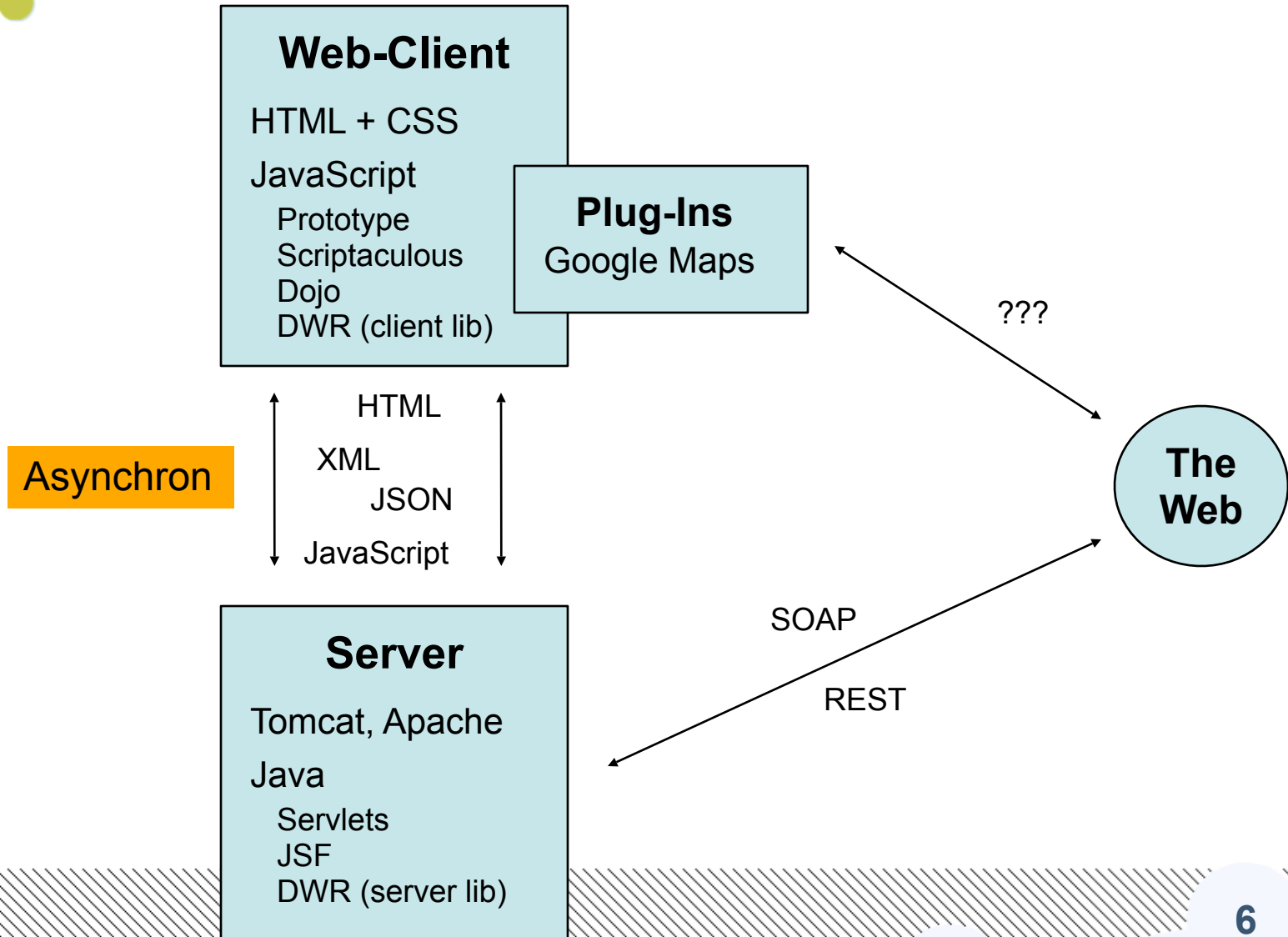
Tomcat, Apache
Java
Servlets
JSF
DWR (server lib)







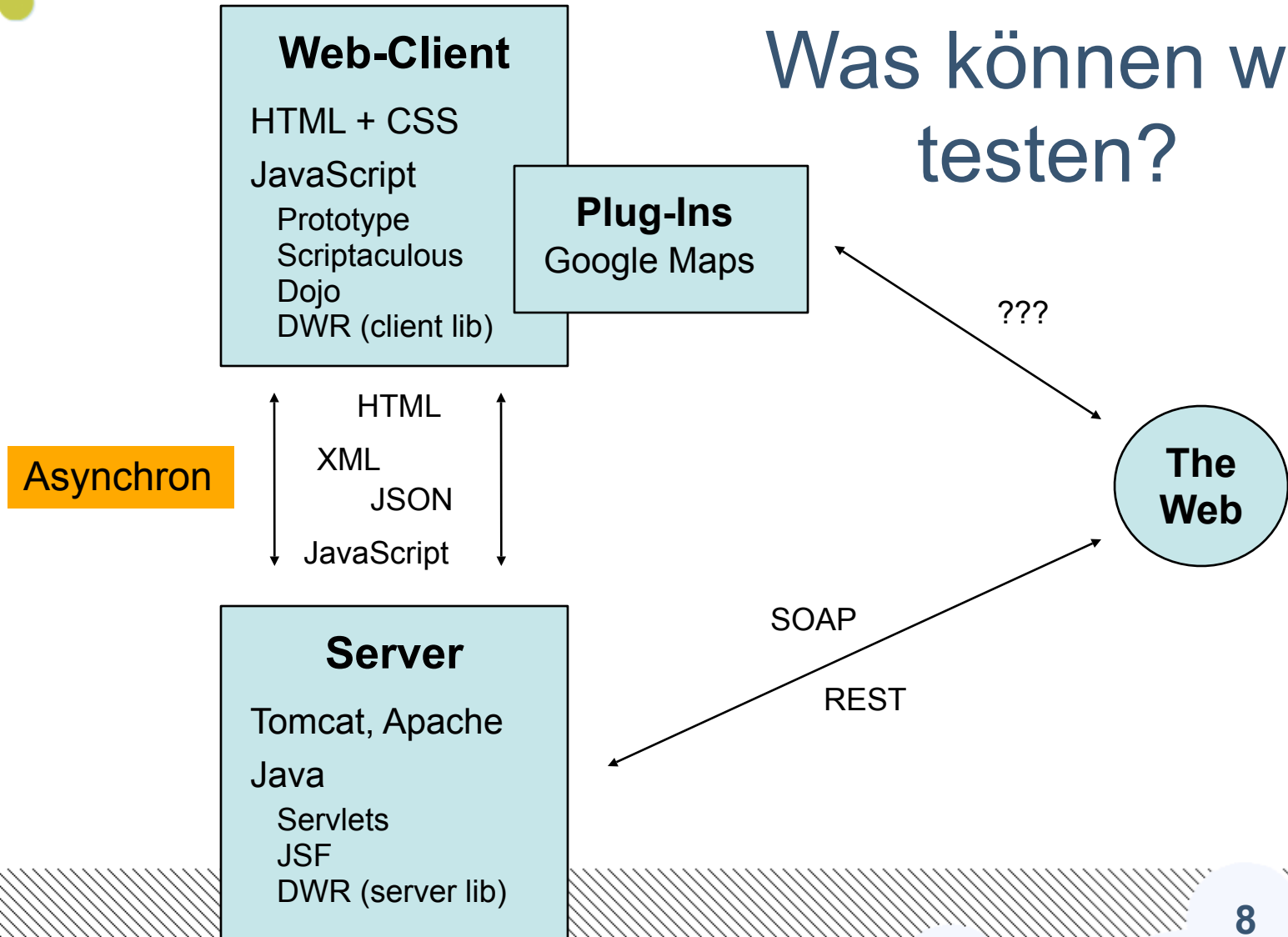


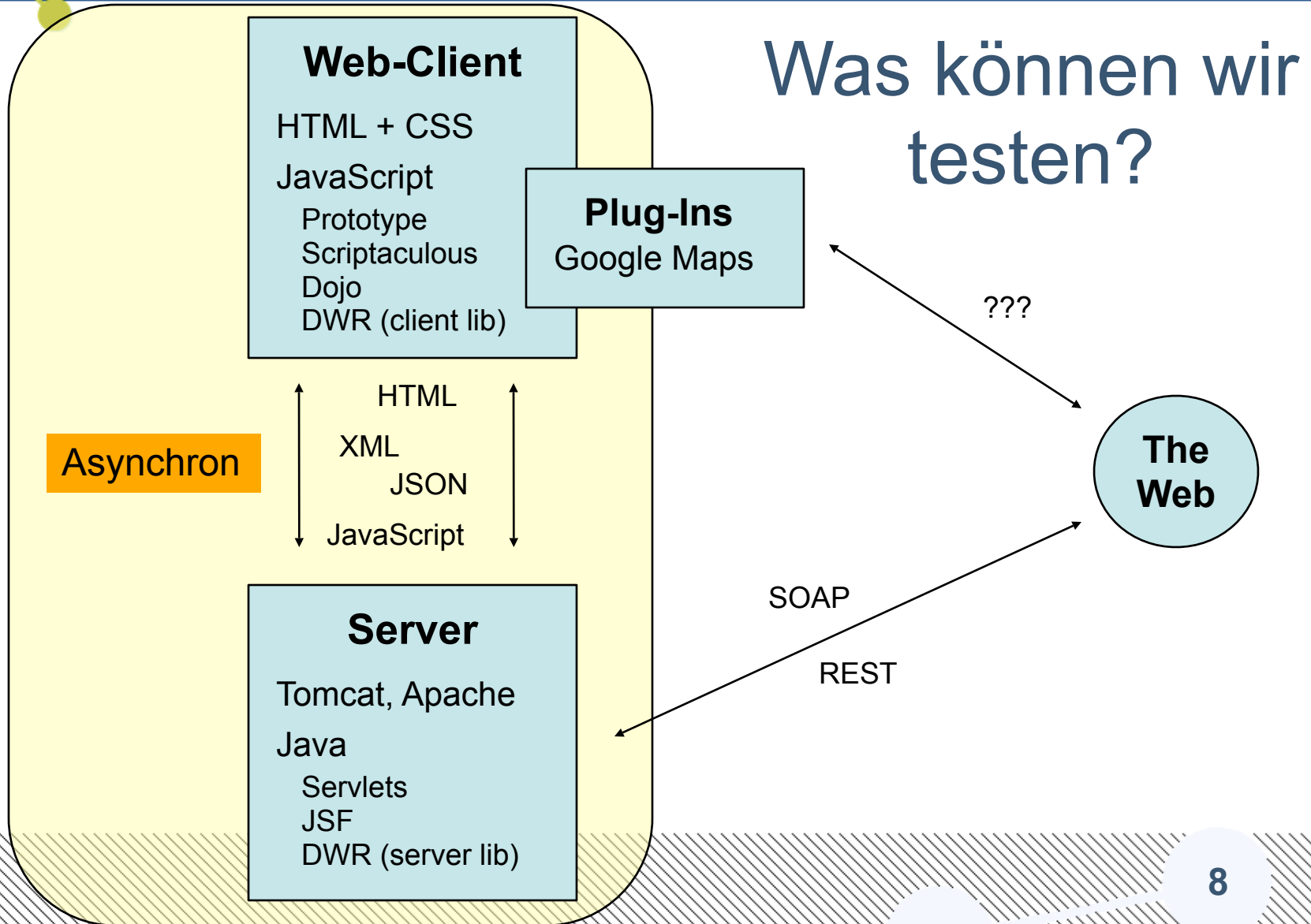


Herausforderungen

- Technologie-Mix (JavaScript, Java etc.)
- Verteilung
- Einbindung entfernter Komponenten
 - Clientseitige „Mashups“ (z.B: Google Maps)
 - Serverseitige „Remote Services“
- Browserinkompatibilitäten

Was können wir testen?





Unit Testing

- Auf dem Server wie gehabt
- Auf dem Client
 - Unit Tests für JavaScript
 - Attrappen für APIs nach außen
 - Vermeidung von selbstgeschriebenem JavaScript (z.B. durch GWT, Cross-Compiler)

Unit Tests mit JavaScript

- Im Browser
 - Scriptaculous Testing Framework
 - JUnit
 - J3Unit
- Browserunabhängig
 - Rhino als JavaScript-Engine
 - JsTester
- Attrappen
 - für Ajax-Kommunikation
 - für Mashup-Komponenten

Scriptaculous Testing Framework

- <http://wiki.script.aculo.us/scriptaculous/show/UnitTesting>
- Testfälle werden auf HTML-Seiten definiert und ausgeführt
- Vorteile:
 - Browserinkompatibilitäten werden sichtbar
 - Alle Browser-Features sind verfügbar
- Nachteil:
 - Automatisierung erfordert Browser bzw. Browsersimulation

Scriptaculous Test Framework Beispiel:

```
<html>...
  <script src="../../../js/prototype.js" type="text/javascript"></script>
  <script src="../../../js/scriptaculous/unittest.js" type="text/javascript"></script>
<body>
  <!-- Log output -->
  <div id="testlog"></div>

  <!-- here go any elements you do the testing on -->
  <script type="text/javascript" language="javascript">
    function myAdder(op1, op2)
    {
      return op1 + op2;
    }
  </script>

  <!-- Tests -->
  <script type="text/javascript" language="javascript">
// <![CDATA[
  new Test.Unit.Runner({
    setup: function() {
      eins = 1;
      zwei = 2;
    },
    testAddSmallNumbers: function() { with(this) {
      assertEquals(3, myAdder(eins, zwei));
    }},
    ...
  });
// ]]>
</script>
```

Scriptaculous Test Framework Beispiel:

```

<html>...
  <script src="../../../js/prototype.js" type="text/javascript"></script>
  <script src="../../../js/scriptaculous/unittest.js" type="text/javascript"></script>
<body>
  <!-- Log output -->
  <div id="testlor"></div>
  <!--
  <sc
    3 tests, 2 assertions, 1 failures, 0 errors
  </sc
  </s
  <!--
  <sc
  //
  new Test.Unit.Runner({
    setup: function() {
      eins = 1;
      zwei = 2;
    },
    testAddSmallNumbers: function() { with(this) {
      assertEquals(3, myAdder(eins, zwei));
    }},
    ...
  });
  // ]]>
</script>

```

| Status | Test | Message |
|--------|----------------------|--|
| passed | testAddSmallNumbers | 1 assertions, 0 failures, 0 errors |
| passed | testAddLargeNumbers | 1 assertions, 0 failures, 0 errors |
| failed | testAddStringNumbers | 0 assertions, 1 failures, 0 errors Failure: assertEquals: expected "3", actual "12" |

JsTester

- <http://jstester.sourceforge.net/>
- Vorteile:
 - JavaScript-Code kann auf Server-Seite ausgeführt und getestet werden
 - Integration mit JUnit / TestNG
- Nachteile:
 - Browser-Funktionalität steht nicht zur Verfügung
 - Browser-Probleme und Inkompatibilitäten werden nicht entdeckt
- Besonders zu empfehlen, wenn der Server selbst JavaScript-Code generiert

JsMock: <http://jsmock.sourceforge.net/>

Object to mock:

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

Object under test:

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

Unit Test:

```
testSpeaker: function() { with(this) {  
    mockControl = new MockControl();  
    speakerMock = mockControl.createMock  
    (Speaker);  
    speakerMock.expects().say('oopsoops');  
    Speaker = speakerMock;  
  
    DoubleSpeaker.say('oops');  
  
    mockControl.verify();  
}}
```

Kundentests

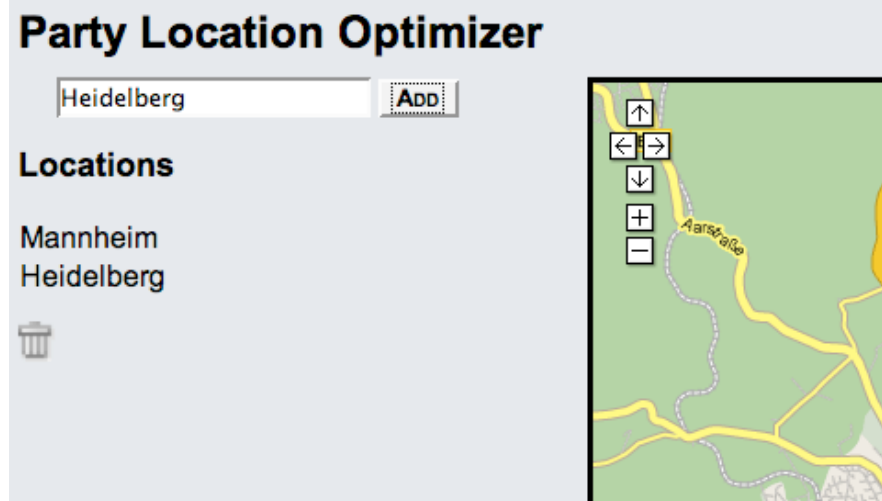
- Möglichst Test aller integrierten Komponenten
- Webclient-Steuerung
 - Im richtigen Browser: Selenium, Watir, Squish ...
 - Browsersimulation: HtmlUnit, Webtest, ...
- Mocking von „Remote Services“
- Zwei Ansätze
 - Oberflächen-orientiert
 - Fachlich orientiert

Oberflächenbasierte Kundentests

- Selenium
 - Der „richtige“ Browser als Testplattform
 - <http://openqa.org/selenium/>
 - Selenium IDE oder Remote Control
- Canoo WebTest
 - Verwendet HtmlUnit (Java-Browser-Simulation)
 - Ant-basierte XML-Skripte oder Groovy
 - <http://webtest.canoo.com/>
- Beide unterstützen JavaScript und Asynchronität

Selenium

| New Test | | |
|-------------------|-------------------|------------|
| open | /plo/ | |
| type | descriptionField | Mannheim |
| click | addLocationButton | |
| type | descriptionField | Heidelberg |
| click | addLocationButton | |
| verifyTextPresent | Heidelberg | |



Canoo WebTest: Ant-Skript

```
<target name="all" depends="clean">
  <webtest name="myTest">
    <config
      host="localhost"
      port="8080"
      protocol="http"
      basepath="" />
    <steps>
      <invoke
        description="get Login Page"
        url="/plo" />
      <setInputField
        description="set a new location"
        htmlId="descriptionField"
        value="Heidelberg" />
      <clickButton
        description="adding..."
        htmlId="addEntryButton" />
      <verifyText description="Check "
        text="Heidelberg" />
    </steps>
  </webtest>
  <formatResults/>
</target>
```

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')  
    }  
}
```

Fachlich orientiertes Framework: Fit / FitNesse

- Framework for Integrated Tests
- Testdaten werden tabellarisch erstellt (in HTML, Excel, MS Word oder im Wiki)
- Sprache des Kunden
- Anbindung ans System in Java
- Portierung für aktuelle Sprachen verfügbar
- <http://fit.c2.com>

FIT...

1. liest HTML-Dokumente ein,
2. führt die enthaltenen Testfälle aus
3. reichert HTML um Testergebnis an
4. gibt HTML-Dokumente wieder aus

| fit.Action | | |
|-------------------|---|----------|
| start | Uebung7.Loesung.Fit.KontoverwaltungFixture | |
| enter | Kundenname | Link |
| press | Neues Konto | |
| check | Kontonummer | 1 |
| enter | Einzahlung | 55.00 |
| check | Kontostand | 55.00 |
| enter | Einzahlung | 45.00 |
| check | Kontostand | 105.00 |
| enter | Kundenname | Westphal |
| press | Neues Konto | |
| check | Kontonummer | 2 |
| enter | Einzahlung | 75.00 |
| check | Kontostand | 75.00 |

| fit.Action | | fit.Action | |
|------------|-----------------------------|------------|--|
| start | Uebung7.Loesung.Fit.Kont... | start | Uebung7.Loesung.Fit.KontoverwaltungFixture |
| enter | Kundenname | enter | Kundenname Link |
| press | Neues Konto | press | Neues Konto |
| check | Kontonummer | check | Kontonummer 1 |
| enter | Einzahlung | enter | Einzahlung 55.00 |
| check | Kontostand | check | Kontostand 55.00 |
| enter | Einzahlung | enter | Einzahlung 45.00 |
| check | Kontostand | check | Kontostand 105.00 <i>expected</i> |
| enter | Kundenname | check | Kontostand 100.00 <i>actual</i> |
| press | Neues Konto | enter | Kundenname Westphal |
| check | Kontonummer | press | Neues Konto |
| enter | Einzahlung | check | Kontonummer 2 |
| check | Kontostand | enter | Einzahlung 75.00 |
| | | check | Kontostand 75.00 |

Wie verbinde ich Fit mit Ajax?

- Verwendung der Business Facade (Server)
- Steuerung des Client
 - HtmlUnit
 - Mozilla Web Client
 - Selenium Remote Control
 - Und WebTest?


Beispiel: Party Location Optimizer

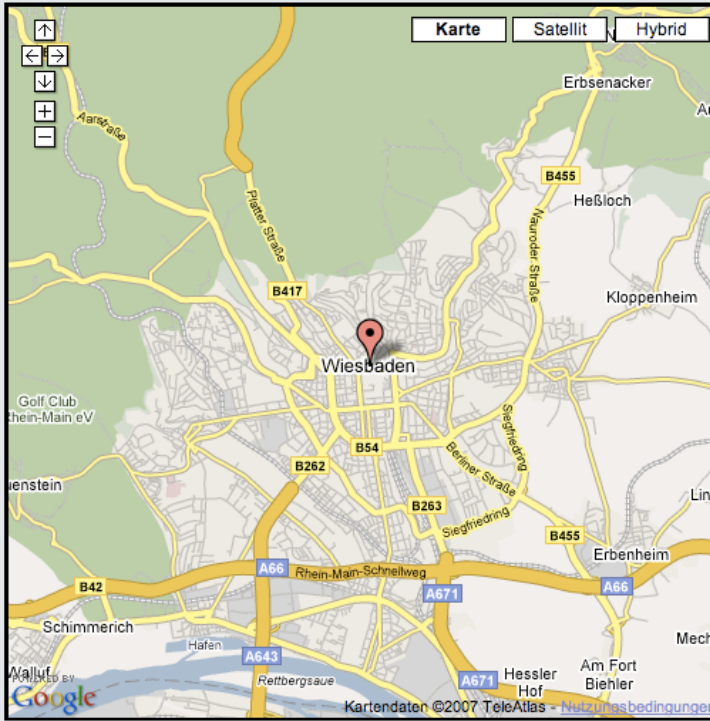
Party Location Optimizer

München

Locations

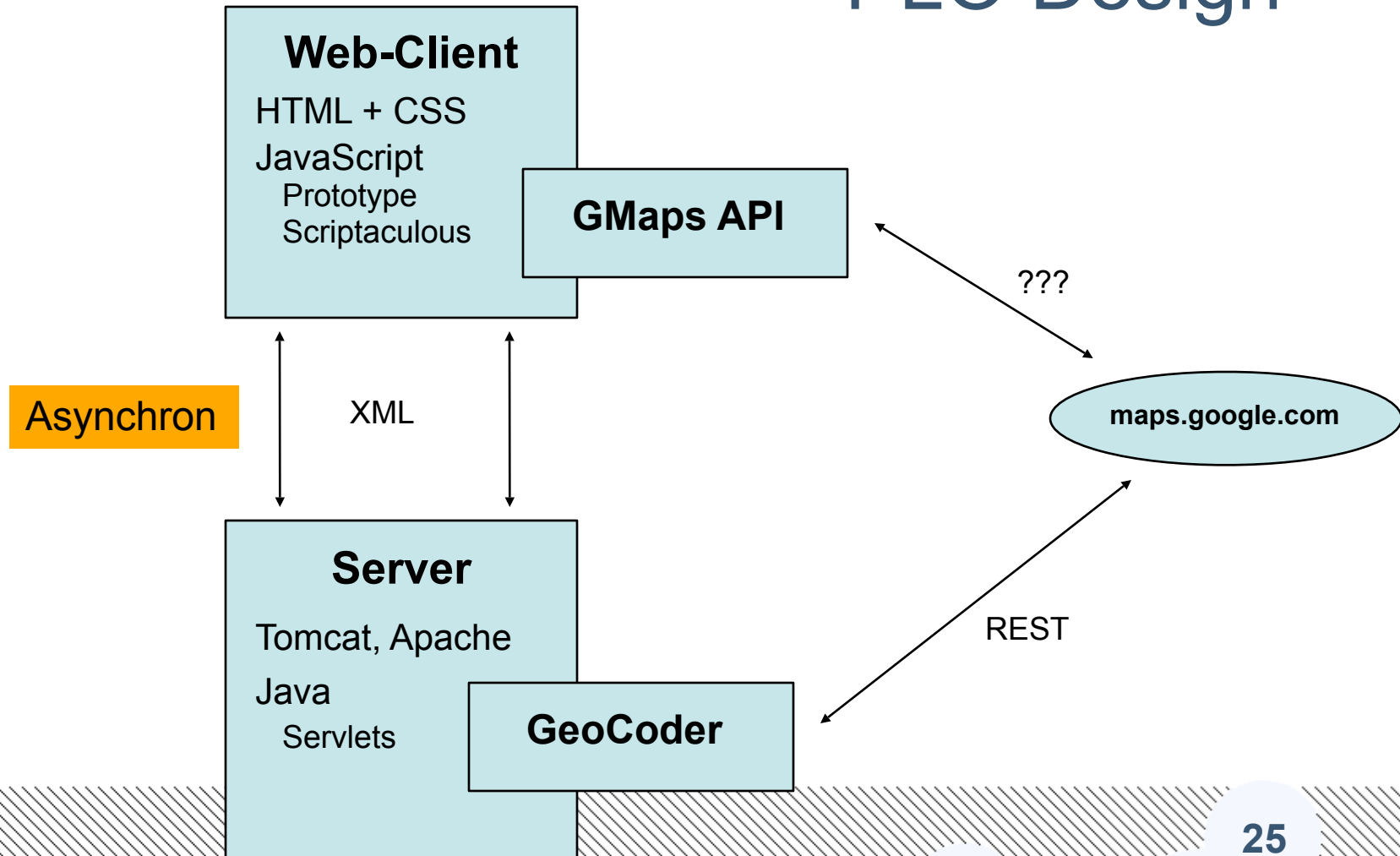
- Wiesbaden
- Frankfurt
- München



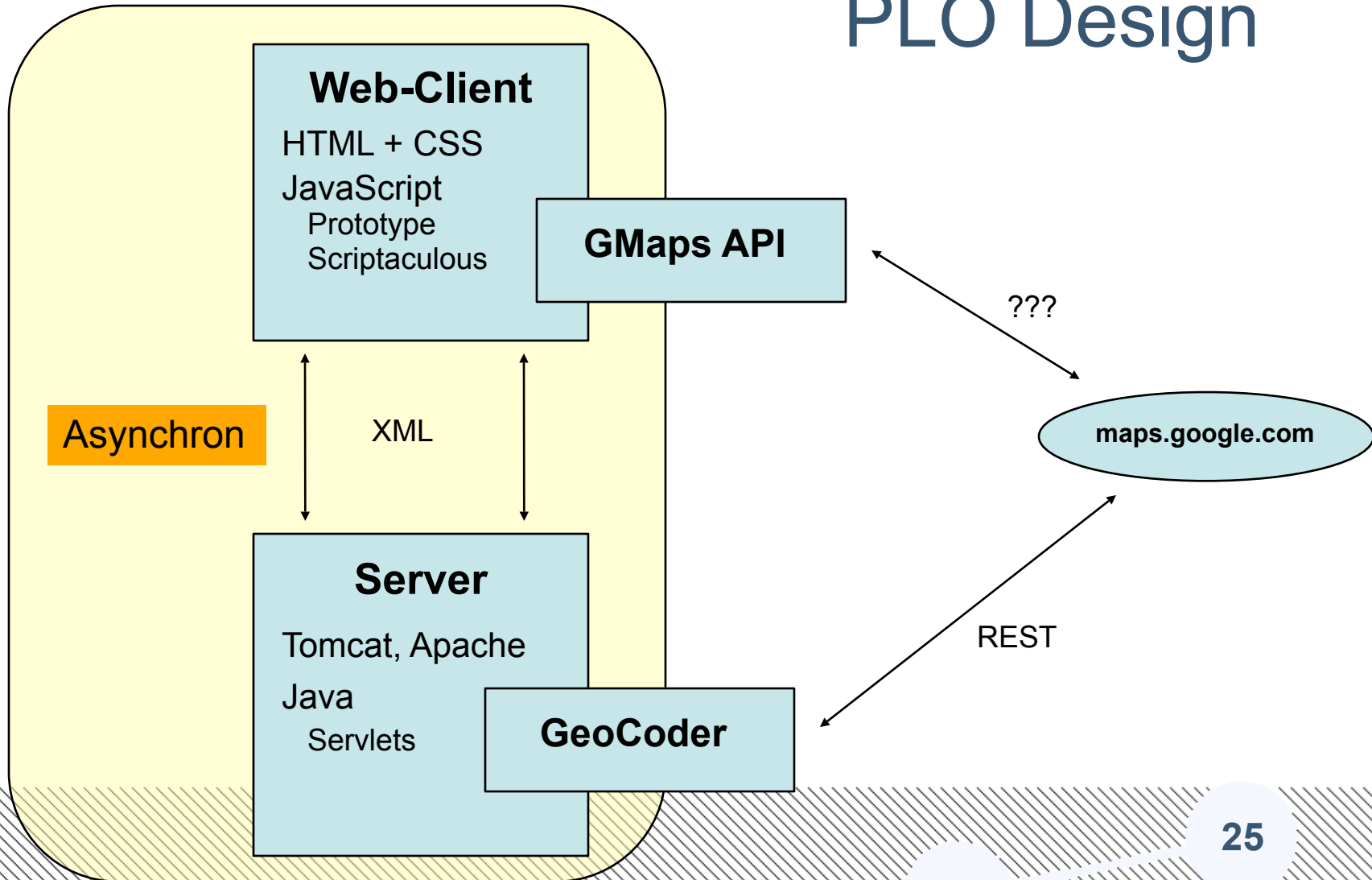


The map shows Wiesbaden with a red pin. Navigation controls include arrows for pan, zoom in (+), zoom out (-), and a 'Karte' button. Map style buttons for 'Karte', 'Satellit', and 'Hybrid' are visible. The map displays roads (A66, A671, B417, B455, B262, B54, B263, B42, A643, A671) and landmarks like 'Golf Club Rhein-Main eV' and 'Rettbergsaue'. A copyright notice at the bottom reads 'Kartendaten ©2007 TeleAtlas - Nutzungsbedingungen'.

PLO Design



PLO Design



Mocken der Google Maps API:

```
if (!GMap2) {  
    var GMap2 = null;  
    var GLatLng = null;  
    ...  
}  
function initGMapMocks() {  
    GLatLng = function(latt, long) {  
        return {latt: latt, long:long};  
    }  
    GMap2.markers = [];  
    GMap2 = function(id) {  
        GMap2.mapId = id;  
        return GMap2;  
    }  
    GMap2.setCenter = function(focus, zoom) {  
        GMap2.point = focus;  
        GMap2.zoomFactor = zoom;  
    }  
}
```

Mocken der Google Maps API:

```
if (!GMap2) {  
    var GMap2 = null;  
    var GLatLng = null;  
    ...  
}  
function initGMapMocks() {  
    GLatLng = function(latt, long) {  
        return {latt: latt, long:long};  
    }  
    GMap2.markers = [];  
    GMap2 = function(id) {  
        GMap2.mapId = id;  
        return GMap2;  
    }  
    GMap2.setCenter = function(focus, zoom) {  
        GMap2.point = focus;  
        GMap2.zoomFactor = zoom;  
    }  
}
```

```
testDisplayMapNoInfo: function() { with(this) {  
    Map.displayMap(10,  
        {long:13.411895,latt:52.523781}, false);  
    assertEquals(10, GMap2.zoomFactor);  
    assertEquals("map", GMap2.mapId);  
    assertEquals(52.523781, GMap2.point.latt);  
    assertEquals(13.411895, GMap2.point.long);  
    assertEquals(0, GMap2.markers.length);  
}}
```

Testen asynchroner Codestellen (1):

```
showMap = function(mapInfo) {  
    if (mapInfo.error) {  
        showErrorText(mapInfo.errorText);  
        return;  
    }  
    Map.displayMap(mapInfo.zoom, mapInfo.focus,  
        mapInfo.showOptimal, mapInfo.places);  
}  
  
function addLocation() {  
    var locationDescription =  
        $('descriptionField').value;  
    Server.addLocation(locationDescription,  
        function(mapInfo) {  
            showMap(mapInfo);  
            if (!mapInfo.error) {  
                addLocationEntry(locationDescription);  
            }  
        })  
}
```

Testen asynchroner Codestellen (1):

```
showMap = function(mapInfo) {  
    if (mapInfo.error) {  
        showErrorText(mapInfo.errorText);  
        return;  
    }  
    Map.displayMap(mapInfo.zoom, mapInfo.focus,  
        mapInfo.showOptimal, mapInfo.places);  
}  
function addLocation() {  
    var locationDescription =  
        $('descriptionField').value;  
    Server.addLocation(locationDescription,  
        function(mapInfo) {  
            showMap(mapInfo);  
            if (!mapInfo.error) {  
                addLocationEntry(locationDescription);  
            }  
        }  
    )  
}
```

```
setup: function() {  
    mockControl = new MockControl();  
    Server = mockControl.createMock(PLOServer);  
    Map = mockControl.createMock(Map);  
}  
testAddLocationError: function() { with(this) {  
    var mapInfo = {error: false, zoom: 3, focus: "f", showOptimal:  
        false, places: []};  
    Server.expects().addLocation("Wiesbaden", TypeOf.isA  
        (Function)).andStub(function() {  
        closure = arguments[1];  
        closure(mapInfo);  
    }  
    );  
    Map.expects().displayMap(mapInfo.zoom, mapInfo.focus,  
        mapInfo.showOptimal, mapInfo.places);  
    $('descriptionField').value = "Wiesbaden";  
    addLocation();  
    mockControl.verify();  
}}
```

Testen asynchroner Codestellen (2):

```
showErrorText = function(text) {  
    Element.update('errorText', text);  
    setTimeout(function() {  
        Effect.Fade('errorText',  
            {afterFinish: function() {  
                Element.update('errorText', "")  
                Element.show('errorText');  
            }}  
        );  
    }, 5000);  
}
```

Testen asynchroner Codestellen (2):

```
showErrorText = function(text) {  
    Element.update('errorText', text);  
    setTimeout(function() {  
        Effect.Fade('errorText',  
            {afterFinish: function() {  
                Element.update('errorText', "")  
                Element.show('errorText');  
            }}  
        );  
    }, 5000);  
}
```

```
testErrorTextAppears: function() { with(this) {  
    assertNull($("#errorText").firstChild);  
    showErrorText("error text");  
    assertEquals("error text", $  
        ("errorText").firstChild.nodeValue);  
}}  
  
testErrorTextDisappears: function() { with(this) {  
    showErrorText("error text");  
    setTimeout(function() {  
        assertNull($("#errorText").firstChild);  
    }, 6000);  
}}
```

Testen asynchroner Codestellen (2):

```
showErrorText = function(text) {  
    Element.update('errorText', text);  
    setTimeout(function() {  
        Effect.Fade('errorText',  
            {afterFinish: function() {  
                Element.update('errorText', "")  
                Element.show('errorText');  
            }}  
        );  
    }, 5000);  
}
```

```
testErrorTextAppears: function() { with(this) {  
    assertNull($("#errorText").firstChild);  
    showErrorText("error text");  
    assertEquals("error text", $  
        ("errorText").firstChild.nodeValue);  
}}  
  
testErrorTextDisappears: function() { with(this) {  
    showErrorText("error text");  
    setTimeout(function() {  
        assertNull($("#errorText").firstChild);  
    }, 6000);  
}}
```

Testen asynchroner Codestellen (2):

```
showErrorText = function(text) {  
    Element.update('errorText', text);  
    setTimeout(function() {  
        Effect.Fade('errorText',  
            {afterFinish: function() {  
                Element.update('errorText', '')  
                Element.show('errorText');  
            }}  
        );  
    }, 5000);  
}
```

```
testErrorTextAppears: function() { with(this) {  
    assertNull($(".errorText").firstChild);  
    showErrorText("error text");  
    assertEquals("error text", $  
        ("errorText").firstChild.nodeValue);  
}}  
  
testErrorTextDisappears: function() { with(this) {  
    showErrorText("error text");  
    setTimeout(function() {  
        assertNull($(".errorText").firstChild);  
    }, 6000);  
}}
```

Mocken der Prototype Ajax-Aufrufe:

```
new Ajax.Request("http://server/ajax", {
  method: 'post',
  postBody: '<plo-req>...</plo-req>',
  contentType: 'text/xml',
  onComplete: function(transport) {
    ...
  }.bind(this)
});
setup: function() {
  mockControl = new MockControl();
  mapInfo = null;
  callback = function() {
    mapInfo = arguments[0];
  }
  Ajax = mockControl.createMock(Ajax);
  url = "http://myserver/ajaxcall";
```

Mocken der Prototype Ajax-Aufrufe:

```
new Ajax.Request("http://server/ajax", {
  method: 'post',
  postBody: '<plo-req>...</plo-req>',
  contentType: 'text/xml',
  onComplete: function(transport) {
    ...
  }.bind(this)
});
setup: function() {
  mockControl = new MockControl();
  mapInfo = null;
  callback = function() {
    mapInfo = arguments[0];
  }
  Ajax = mockControl.createMock(Ajax);
  url = "http://myserver/ajaxcall";
```

```
testGetInitialMap: function() { with(this) {
  Ajax.expect().Request(url, TypeOf.isA(Object)).
  andStub(function() {
    onComplete = arguments[1].onComplete;
    onComplete({responseXML: createAnswerDom(
      {long: "1.1", latt:"2.2"}, 5)
    });
  });
  server.getInitialMap(callback);
  assertEquals(5, mapInfo.zoom);
  assertEquals(1.1, mapInfo.focus.long);
  assertEquals(2.2, mapInfo.focus.latt);
  assertEquals(false, mapInfo.showOptimal);
  assertEquals(0, mapInfo.places.length);
  mockControl.verify();
}}
```

AddLocationFixture

```
public class AddLocationFixture extends Fixture {
```

```
    @Override
```

```
    public void doRow(Parse row) {
```

```
        Parse column = row.parts;
```

```
        try {
```

```
            SystemUnderTest.getPLOFacade().addLocation(column.text());
```

```
            right(column);
```

```
        } catch (Exception e) {
```

```
            exception(column, e);
```

```
        }
```

```
    }
```

Selenium Facade – addLocation

```
public class PartyLocationOptimizerFacadeSelenium
    implements PartyLocationOptimizerFacade {

    private Selenium selenium = null;

    public void initialize(String browser, String url) {
        selenium = new DefaultSelenium("localhost", 4444, browser, url);
        selenium.start();
        selenium.open(url);
    }

    public void addLocation(String location) {
        selenium.type("descriptionField", location);
        selenium.click("addLocationButton");
    }
}
```

Selenium Facade – Prüfungen

```
public List<Location> getLocations() {  
    ArrayList<Location> locations = new ArrayList<Location>();  
    for (int i = 1; true; i++) {  
        String location = selenium.getText("xpath=//ul[@id='locations']/li[" + i + "]);  
        [...]  
    }  
    return locations;  
}
```

```
public String getErrorText() {  
    return selenium.getText("id=errorText");  
}
```

```
public GoogleMap getGoogleMap() {  
    return new GoogleMap(selenium.getValue("name=zoom"), [...]);  
}
```

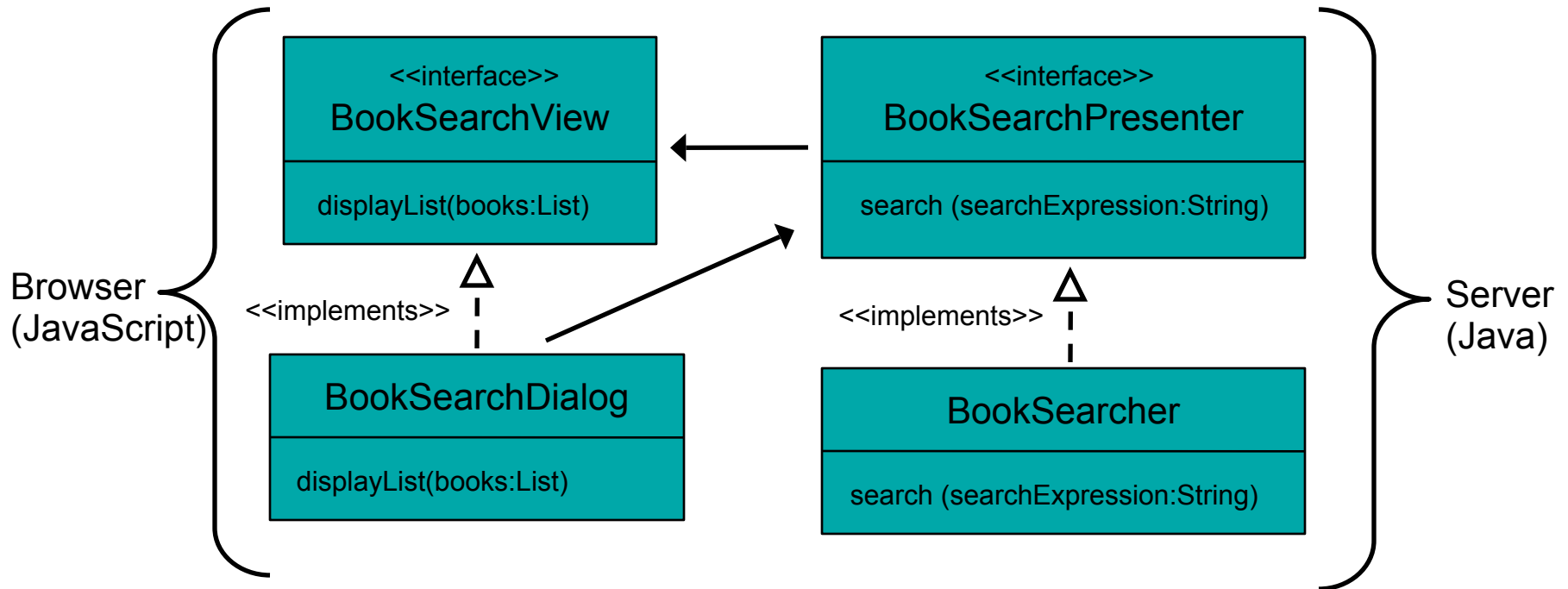
Selenium Facade – Drag&Drop

```
public void removeLocation(String location) {  
  
    // Drag'n'Drop auf den großen Mülleimer  
    String objectToBeDragged = "xpath=//ul[@id='locations']//li[span/text()  
        =',, + location + "']";  
    String dragDestObject = "id=trashcan";  
  
    selenium.dragAndDropToObject(objectToBeDragged, dragDestObject);  
  
    selenium.waitForCondition("!selenium.isVisible(\"id=addIndicator\")",  
        "5000");  
}
```

Ajax- und JavaScript-Debugging

- Firefox: Firebug und Venkman
- Safari: WebInspector und Drosera
- Microsoft Script Debugger für IE
- Eclipse ATF noch „instabil“

Model-View-Presenter mit Ajax?



Ajax-MVP-Framework: <http://sf.net/projects/jayjax>

Fazit

- Testgetriebenes Ajax ist möglich
- Teste im Großen und im Kleinen
- Manuelles Testen wird nicht überflüssig
- Teste auf allen zu unterstützenden Browsern
- Schwer testbare Ajax-Features:
 - Asynchrone Ereignisse
 - Drag N Drop
 - Dynamisch eingeblendete Elemente

Quellen & Referenzen

- <http://mir.aculo.us/stuff/AdventuresInJavaScriptTesting.pdf>
- <http://ajaxian.com/by/topic/testing/>
- http://ajaxpatterns.org/Browser-Side_Test
- <http://jlink.blogger.de/stories/414112/>