

Kai Jäger

Introduction

to **AJAX** *and* **JavaScript**

Presented at **Web 2.0 Day**, Stuttgart Media University on December 1st, 2006
by Kai Jäger » jaeger@xwrs.net » <http://www.kaijaeger.com>

|| | | ||]C |
HOCHSCHULE DER MEDIEN


medien
informatik

M,P,NEWMEDIA,

Agenda

1. *Why do we need* **AJAX?**

Agenda

2. How *does it work?*

Agenda

3. *A short intro to JavaScript.*

Agenda

4. *Understanding* web services.

Agenda

5. *Why use frameworks?*

Agenda

6. *How do I* get started?

I CALCULATED THE TOTAL
TIME THAT HUMANS
HAVE WAITED FOR WEB
PAGES TO LOAD...



S. Adams E-mail: SCOTTADAMS@AOL.COM

IT CANCELS OUT ALL
THE PRODUCTIVITY
GAINS OF THE INFOR-
MATION AGE.



5/5/97 © 1997 United Feature Syndicate, Inc.

Attend web 2.0 day

~~Get laundry~~

Wash car

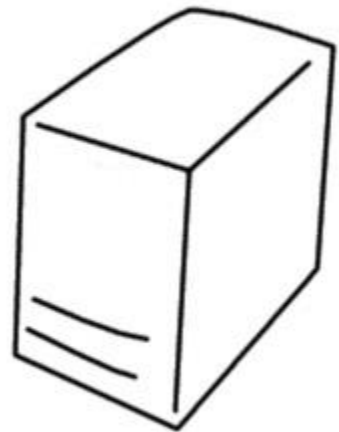
Pay bills

Buy groceries



client

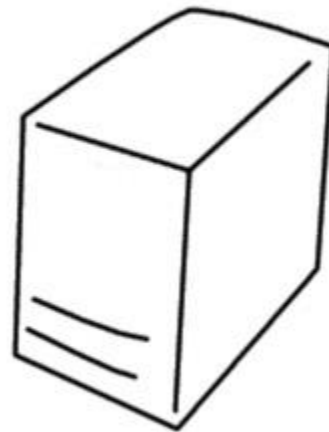
request



web server



client

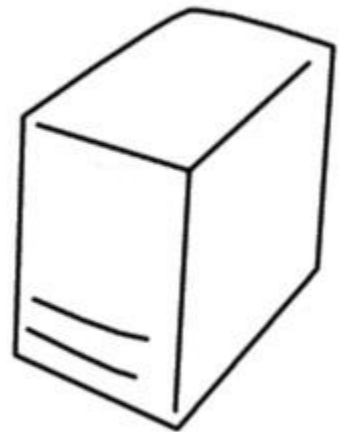


web server



client

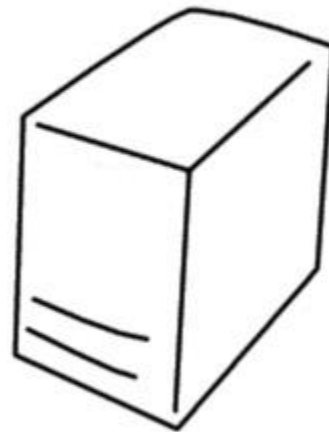
request



web server



client



web server

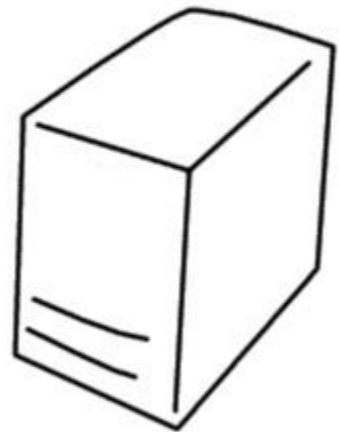
AJAX

is **asynchronous**



client

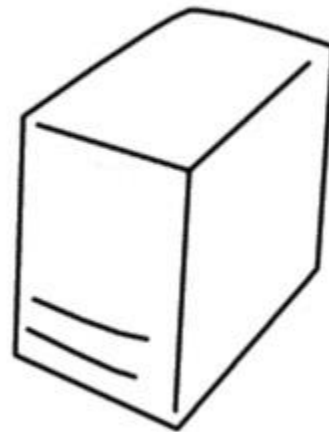
request



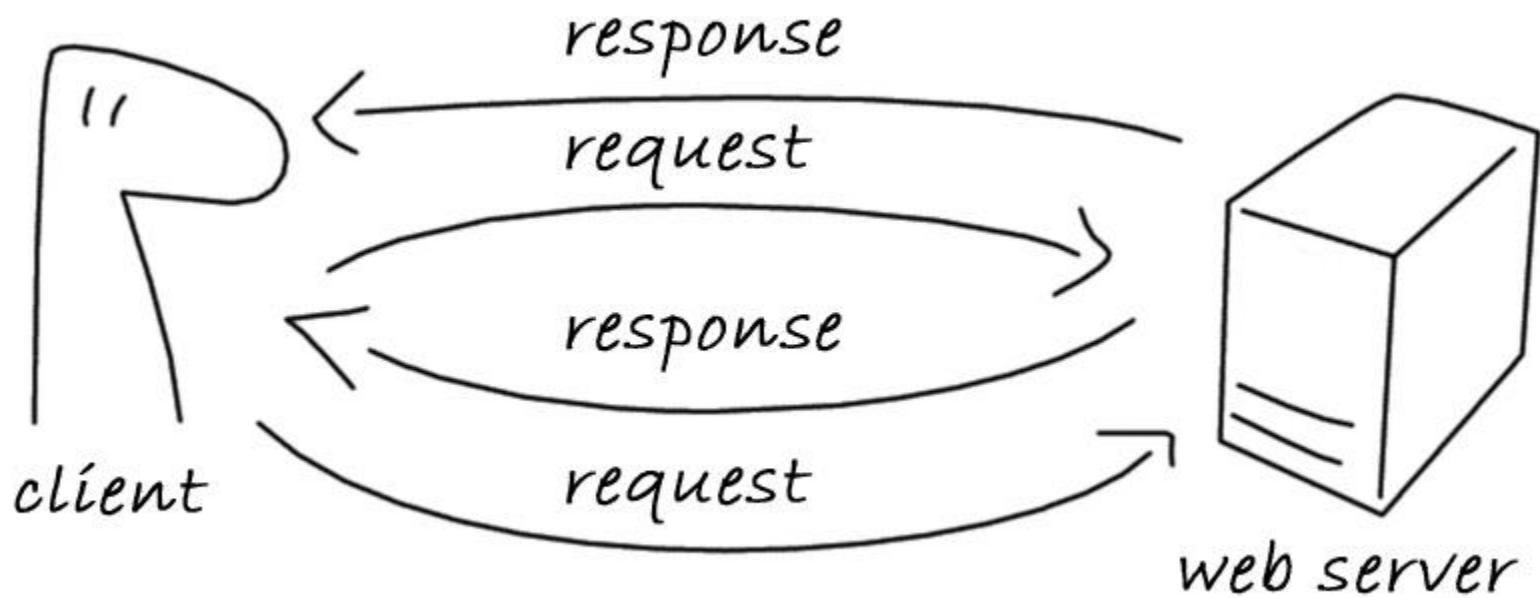
web server



client



web server



AJAX solves *just this one*
problem



user



user



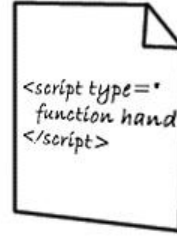
*interacts with
website*



user



interacts with
website



```
<script type="*  
function hand  
</script>
```

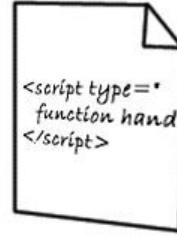
event is
handled by
JavaScript



user



interacts with
website



event is
handled by
JavaScript



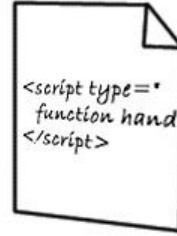
JavaScript
creates a
message in
XML or some
other format



user



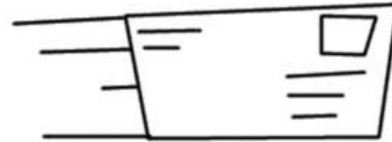
interacts with
website



event is
handled by
JavaScript



JavaScript
creates a
message in
XML or some
other format



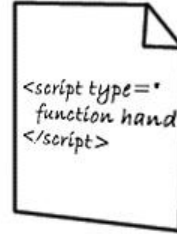
The message
is sent to a
webserver
via HTTP



user



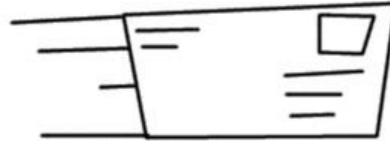
interacts with
website



event is
handled by
JavaScript



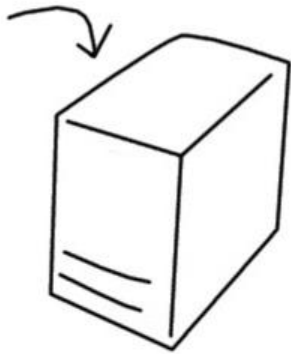
JavaScript
creates a
message in
XML or some
other format



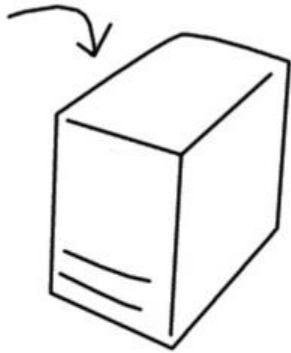
The message
is sent to a
webserver
via HTTP



The user
interface is
updated
immediately



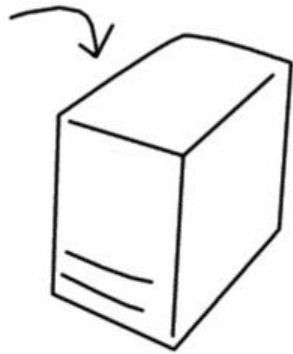
web server
receives and
processes
message



web server
receives and
processes
message



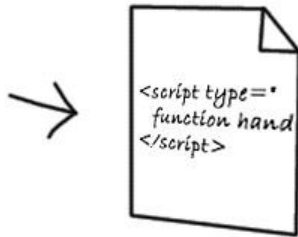
server responds
to message
also in XML
or other format



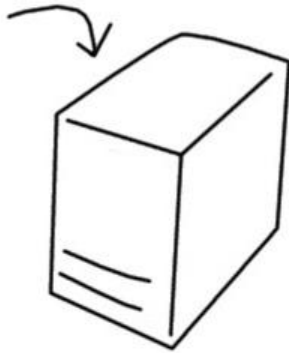
web server
receives and
processes
message



server responds
to message
also in XML
or other format



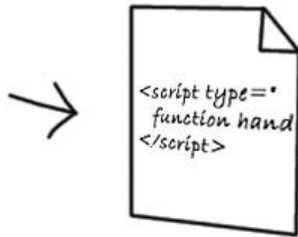
response
invokes a
call-back



web server
receives and
processes
message



server responds
to message
also in XML
or other format



response
invokes a
call-back



JavaScript may
update UI again.

*Basically, what **AJAX**
means is "JavaScript now
works." .. **Paul Graham***

AJAX

relies on **JavaScript**

JavaScript

*is to **Java** what *VBScript* is
to **VisualBasic***



...not!

JavaScript *has more in common with functional languages like Lisp or Scheme than with C or Java.*

.. **Douglas Crockford (Yahoo!)**

Say "hello", the JavaScript way...

```
var salutation = new Array();  
salutation[0] = "Good morning";  
salutation[1] = "Good evening";
```

```
function saySomething(message) {  
    document.write(message);  
}
```

```
var hours = (new Date()).getHours();
```

```
if (hours > 5 && hours < 19) {  
    saySomething(salutation[0]);  
} else {  
    saySomething(salutation[1]);  
}
```

how is that **not**

Java - Script?

Try to do that in Java...

```
function sayHelloTo(name) {  
    var salutation = "Hello ";  
    return function() {  
        document.write(salutation + name);  
    }  
}
```

```
var helloBob = sayHelloTo("Bob");  
var helloJoe = sayHelloTo("Joe");
```

```
helloBob();  
helloJoe();
```

How is that useful?

```
// Event handling
```

```
button.onclick = function() {  
    alert("The button has been clicked");  
}
```

```
// Callbacks
```

```
pendingRequest.oncomplete = function(response) {  
    if (response.error) {  
        alert("Request failed.");  
    }  
}
```

Generic functions

```
function numberCompare(a, b) { ... }
```

```
function alphaCompare(a, b) { ... }
```

```
function alphaCompareNoCase(a, b) { ... }
```

```
function sort(compareFunc, array) { ... }
```

```
var sorted = sort(alphaCompareNoCase, ["W",  
    "e", "b", "T", "w", "o"]);
```

functions +

closures =

objects

How it's done in Java

```
public class Person {  
    private String fullName;  
  
    public Person(String firstname, String lastname) {  
        fullName = firstname + " " + lastname;  
    }  
  
    public String getFullName() {  
        return fullName;  
    }  
}
```

How it's done in JavaScript

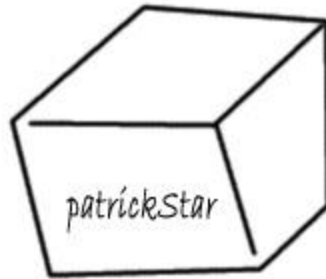
```
function Person(firstname, lastname) {  
    var fullName = firstname + " " + lastname;  
  
    this.getFullName = function() {  
        return fullName;  
    }  
}
```

How it's done in JavaScript

```
function Person(firstname, lastname) {  
    var fullName = firstname + " " + lastname;  
  
    this.getFullName = function() {  
        return fullName;  
    }  
}
```

```
var patrickStar = new Person("Patrick", "Star");
```

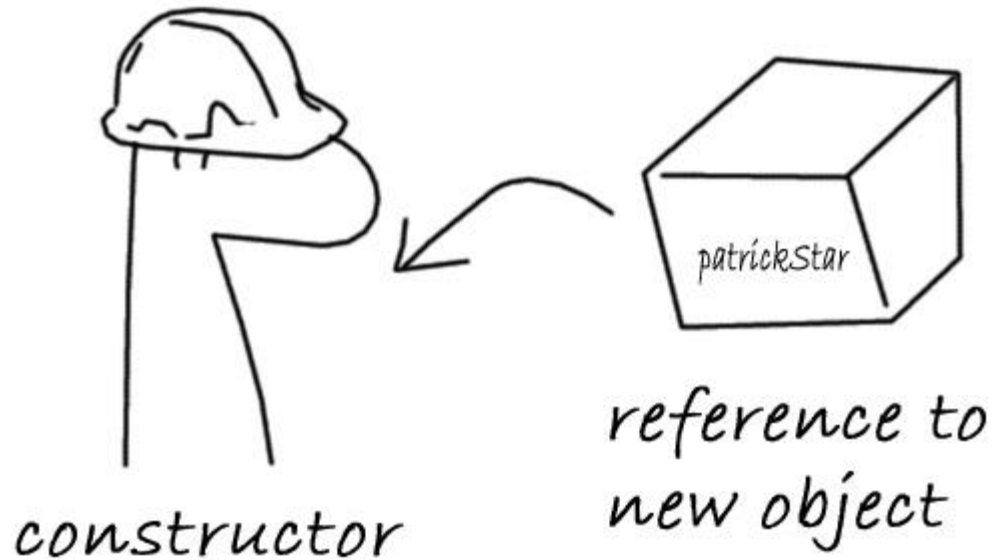
How does it work?



"neutral"
object

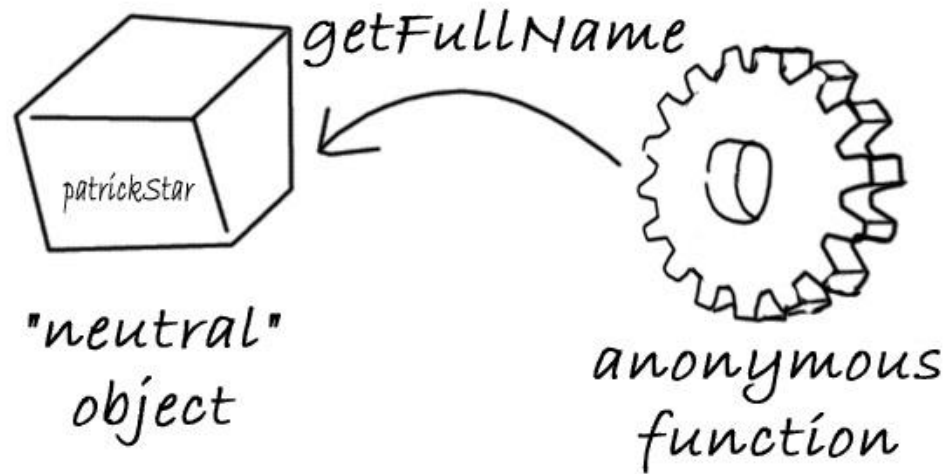
```
var patrickStar = new Object();
```

How does it work?



```
Person.call(patrickStar, "Patrick", "Star");
```

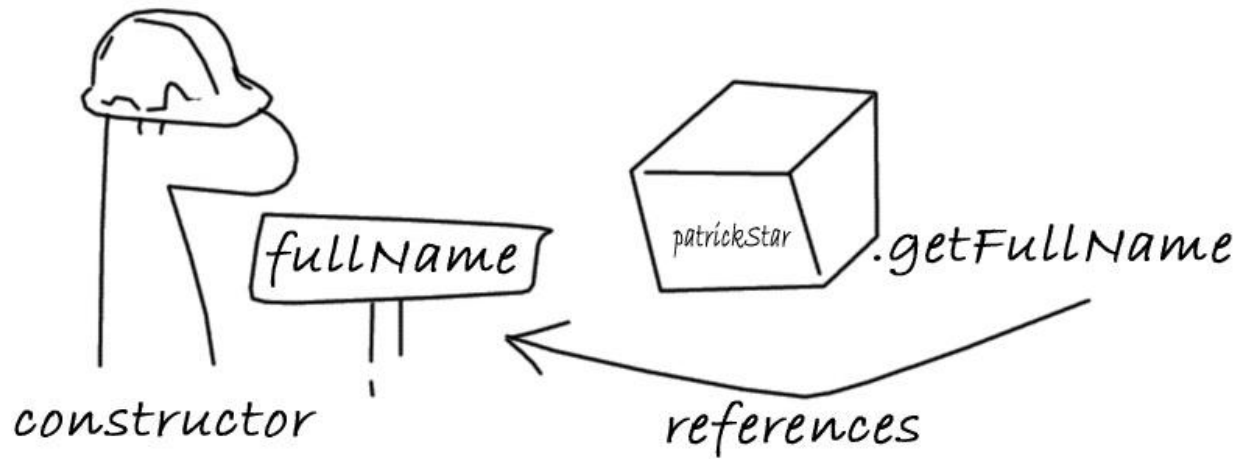
How does it work?



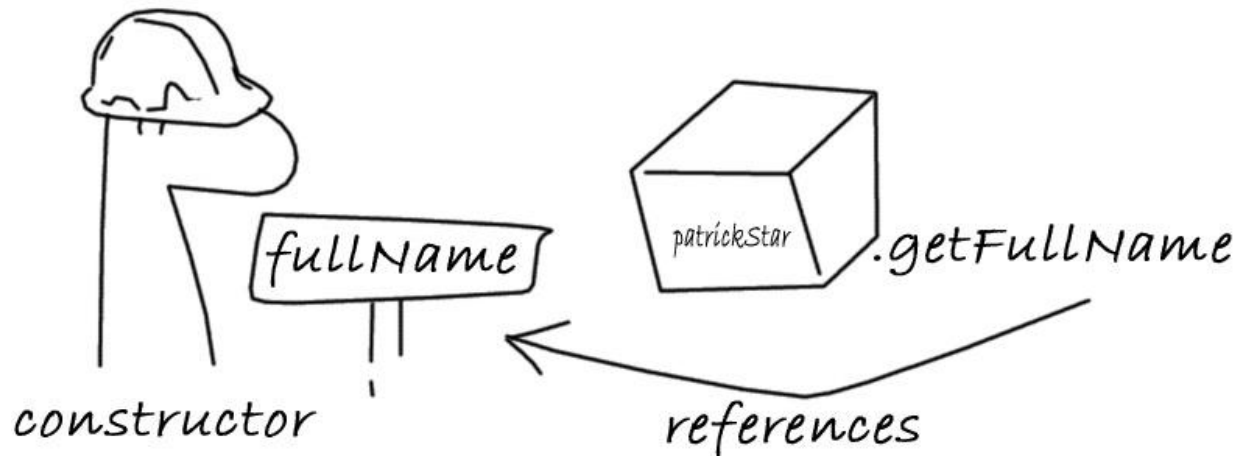
reference to "patrickStar" object

```
this.getFullName = function() {  
    return fullName;  
}
```

How does it work?



How does it work?



*local variable of constructor
becomes private member*

```
this.getFullName = function() {  
    return fullName;  
}
```

AJAX

uses **XML (occasionally)**

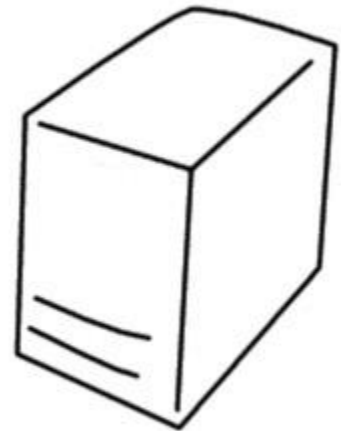


client

<Envelope>
<Body></Body>
</Envelope>

web service protocol

<Envelope>
<Body></Body>
</Envelope>



web server

REST

*REpresentational State
Transfer*

SOAP

*formerly known as Simple
Object Access Protocol.*

Request

```
<?xml version="1.0">  
<env:Envelope xmlns:env="http://.../">  
  <env:Body>  
    <todo:getTasks xmlns:todo="http://.../">  
      <todo:userName>John Doe</todo:userName>  
    </todo:getTask>  
  </env:Body>  
</env:Envelope>
```

Response

```
<?xml version="1.0">
<env:Envelope xmlns:env="http://.../">
  <env:Body>
    <todo:getTasksResponse ...>
      <todo:task>
        <todo:id>1</todo:id>
        <todo:name>Attend Web 2.0
Day</todo:name>
      </todo:task>
    </todo:getTasksResponse>
  </env:Body>
</env:Envelope>
```

XML-RPC

*XML-Remote Procedure
Calls.*

Request

```
<?xml version="1.0">
<methodCall>
  <methodName>todoList.getTasks</methodName>
  <params>
    <param>
      <value><string>John Doe</string></value>
    </param>
  </params>
</methodCall>
```

Response

```
<?xml version="1.0">
<methodResponse>
  <params>
    <param>
      <array>
        <data>
          <value><string>Attend Web 2.0 Day</string></value>
        </data>
        ...
      </array>
    </param>
  </params>
</methodResponse>
```

JSON

JavaScript Object Notation

Request

```
{  
  "method": "getTasks",  
  "params": ["John Doe"]  
}
```

Response

```
{  
  "result": [  
    "Attend Web 2.0 day",  
    "Buy groceries",  
    "Get laundry"  
  ]  
}
```

Alternatives

to XML and JSON

AJAX

putting it all **together**

Here's how it works

```
var xmlHttp = new XMLHttpRequest();
```

Instantiate XHR
(not IE 6 and below)

```
var xmlStr = '<?xml version="1.0"><env:Envelope>...';
```

```
xmlHttp.open("POST", "todo.php", true);  
xmlHttp.onreadystatechange = function() {  
    if (xmlHttp.readyState == 4) {  
        processResponse(xmlHttp.responseXml);  
    }  
}
```

callback function

request method

URL to get/post/delete/put

asynchronous

send the XML to the server

don't *try and do it all*

yourself

*save yourself the **pain**, use a*
framework



YAHOO!
User Interface Library

PLEX



jQuery
New Wave Javascript

QOOXDOD
THE NEW ERA OF WEB DEVELOPMENT

 **ActiveWidgets**

 **BACKBASE**

MACAO

Google
Code
Google Web Toolkit

Rico


developer

 **Zimbra**
Kabuki Ajax Toolkit

SAJAX
SIMPLE AJAX TOOLKIT

script.aculo.us
it's about the user interface, baby!

ASP.net
AJAX

dōjō the javascript toolkit

prototype



Finding the right framework

1. *The big ones are* **prototype and dojo.**

Finding the right framework

2. *Look for documentation / community activity.*

Finding the right framework

3. *Check out sample code and compare coding styles.*

Finding the right framework

4. *See which one integrates best with your **server-side language.***

how do I get

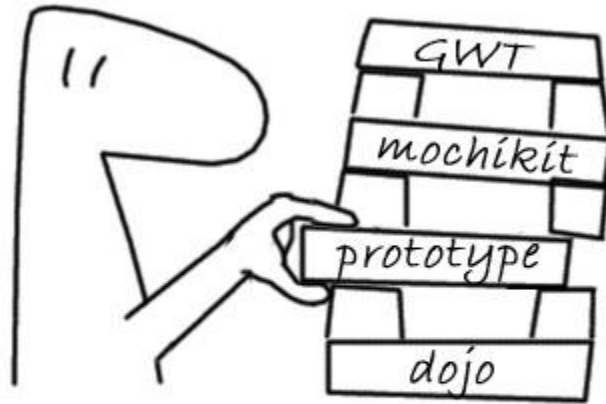
started?



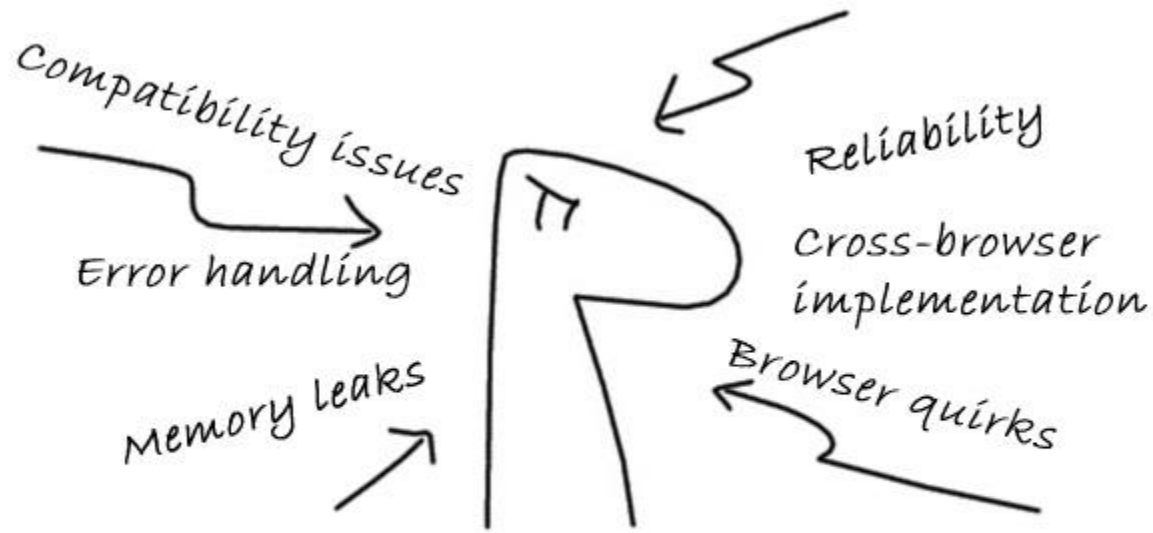
*you (somewhat
simplified)*



Learn
JavaScript



pick an AJAX
framework



or write one yourself



*come up with a
great idea*



*spend endless
nights coding*



*and finally get
acquired by google*

Please *check out my* **blog** *at*
www.kaijaeger.com

AJAX in der Praxis

(working title)

Coming **September 2007***, *published*
by **Springer-Verlag**

** rough estimate*

Thank you

you've been a terrific

audience.