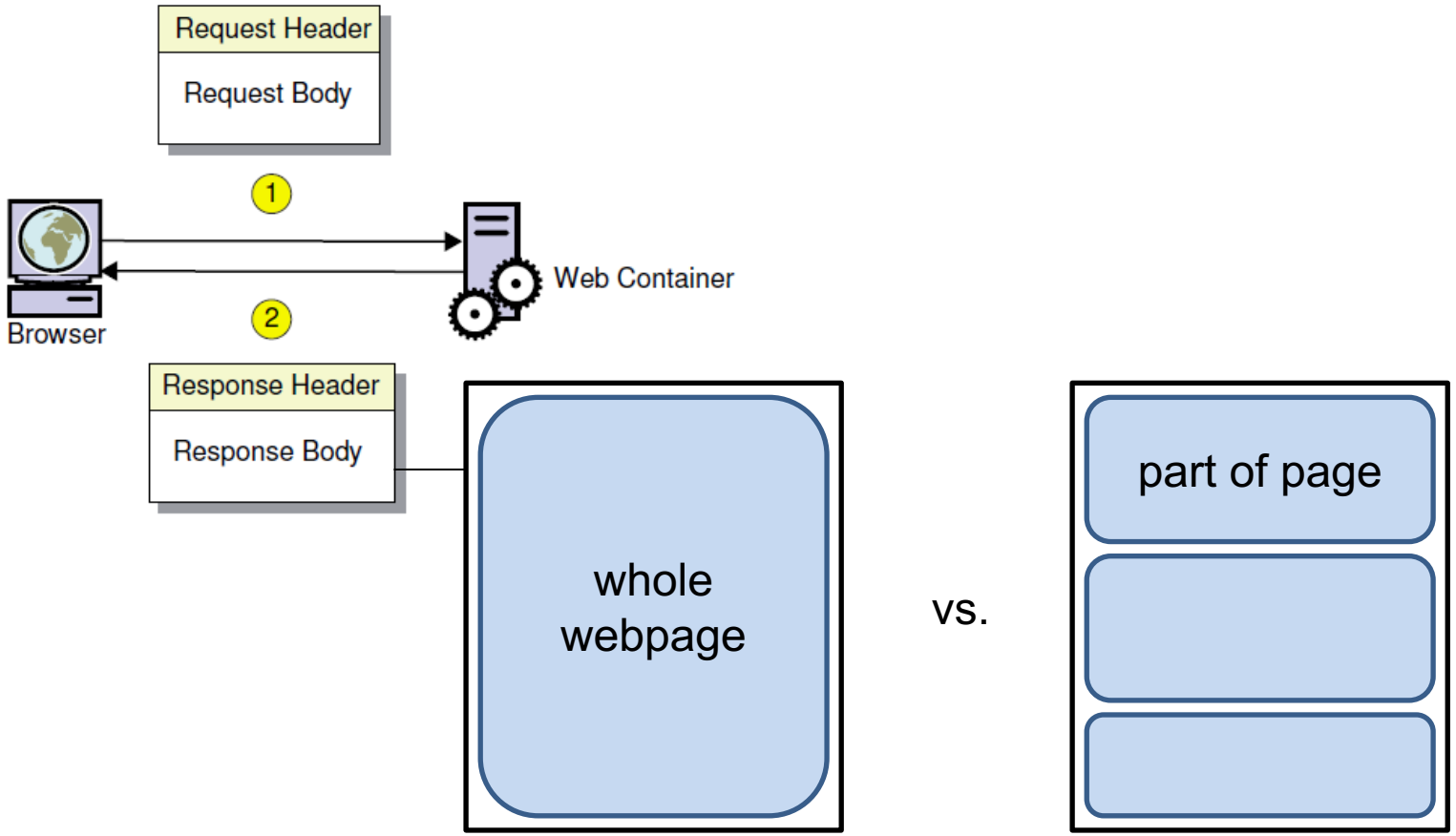
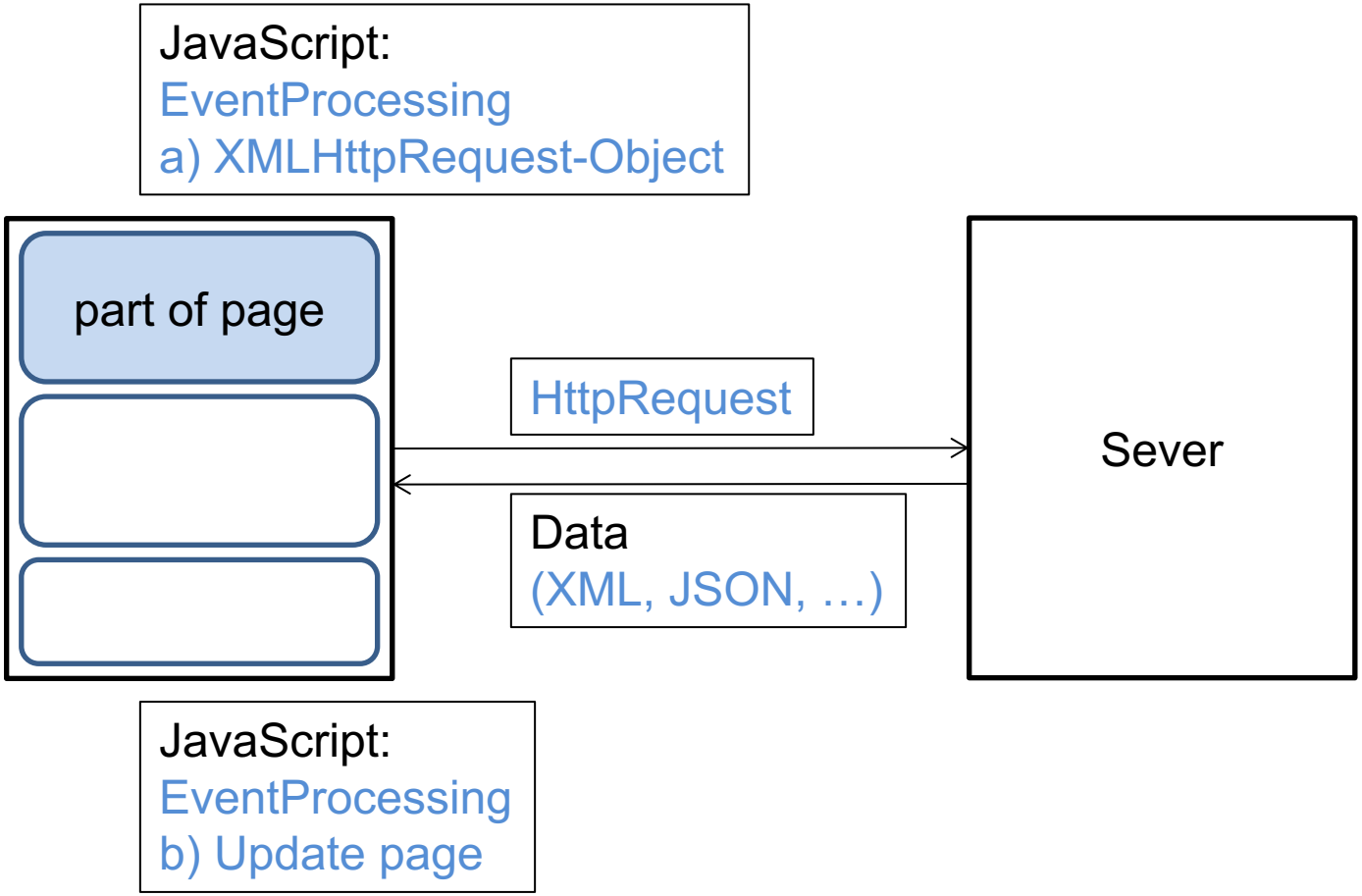
Asynchronous JavaScript and XML (AJAX)

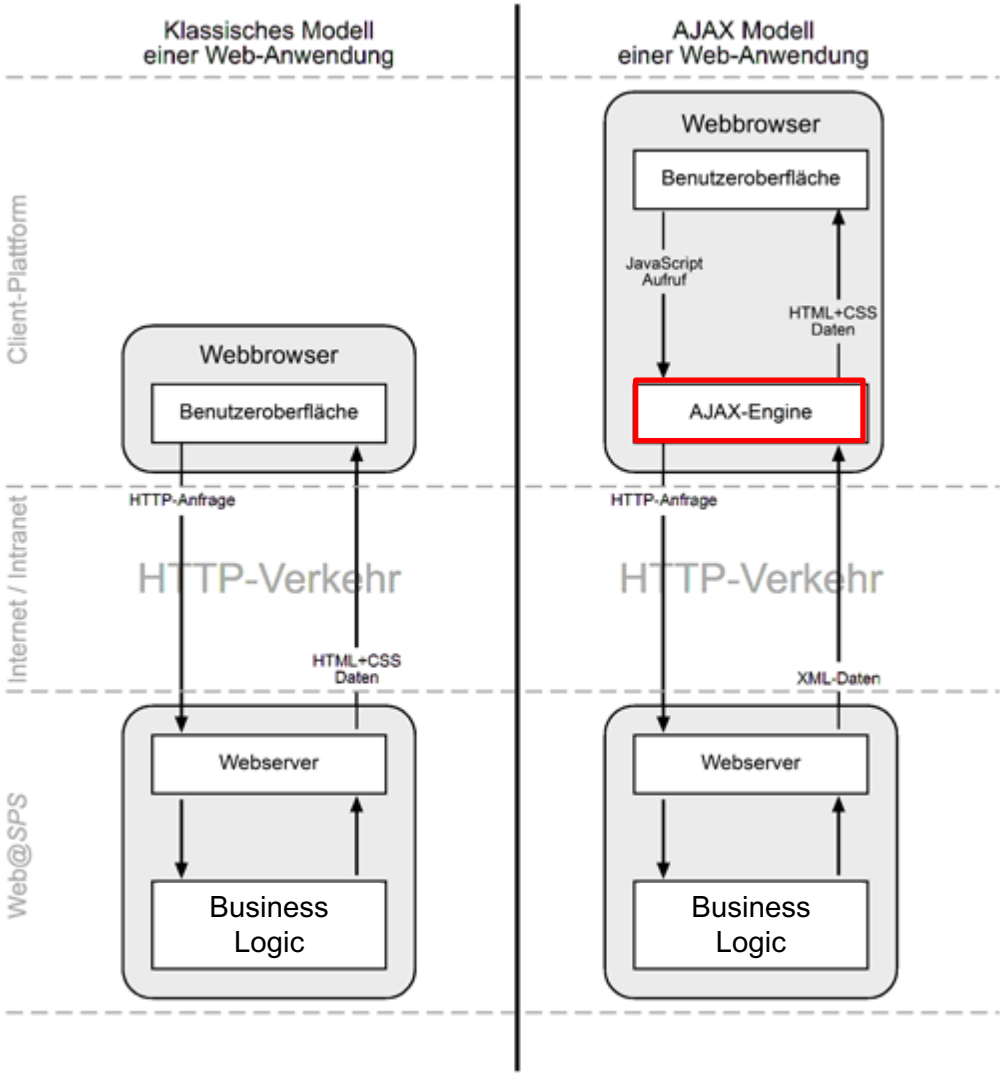
Modul 6
Motivation



Modul 6
Solution



Modul 6
Solution



jQuery
write less, do more.

prototype

mooTools

dojo
toolkit

…

or without …


jQuery
write less, do more.

Modul 6
Motivation

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<script type='text/javascript' src='dwr/engine.js'></script>
<script type='text/javascript' src='dwr/interface/RemoteFunctions.js'></script>

<script>
    function getMessageFromServer()
    {
        ...
    }
</script>

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body onClick="getMessageFromServer();">
        <h1 id="myTag">Hello World!</h1>
    </body>

</html>
```

Mixing layout and logic

jQuery Syntax

The jQuery syntax is tailor made for **selecting** HTML elements and performing some **action** on the element(s).

Basic syntax is: `$(selector).action()`

A **$** sign to define/access jQuery
A (*selector*) to "query (or find)" HTML elements
A jQuery *action*() to be performed on the element(s)

Examples:
`$(this).hide()` - hides the current element.
`$("p").hide()` - hides all <p> elements.
`$(".test").hide()` - hides all elements with class="test".
`$("#test").hide()` - hides the element with id="test".

Quelle: http://www.w3schools.com/jquery

The Document Ready Event

```
$(document).ready(function(){

   // jQuery methods go here...

});
```

or (equivalent)

```
$(function(){

   // jQuery methods go here...

});
```

Quelle: http://www.w3schools.com/jquery

jQuery Selectors

Select all <h> elements on a page

```
$("h1").hide();
```

The #id Selector

Find an element with a specific id

```
$("#myID").hide();
```

Quelle: http://www.w3schools.com/jquery

Modul 6
jQuery Events

Common Events

Mouse Events	Keyboard Events	Form Events	Document/Window Events
click	keypress	submit	load
dblclick	keydown	change	resize
mouseenter	keyup	focus	scroll
mouseleave		blur	unload

jQuery Syntax For Event Methods

```
$("h1").click( function(){
  // action goes here!!
  // e.g. alert("Test-Action") or $(this).hide(); or …
});
```

Quelle: http://www.w3schools.com/jquery

Modul 6
jQuery Set Content

Set Content - text(), html(), and val()

Three methods from to **set content**:

- `text()` - Sets or returns the text content of selected elements
- `html()` - Sets or returns the content of selected elements (including HTML markup)
- `val()` - Sets or returns the value of form fields

```
$("h1").click(function(){
  $("#myID").text("Hello world!");
});

$("h1").click(function(){
  $("#myID").html("<b>Hello world!</b>");
});

$("h1").click(function(){
  $("#myID"). val("Dolly Duck");
});
```

Quelle: http://www.w3schools.com/jquery

Set & Get Attributes - attr()

jQuery attr() method is used to get & set attribute values.

```
$("h6").click(function(){
  alert( $(this).attr("id") );
});



$("#myID").click(function(){
  $(this).attr("style", "background-color:rgb(200, 200, 200);");
});
```

Quelle: http://www.w3schools.com/jquery

Modul 6
jQuery

```
…
<script src="//ajax.aspnetcdn.com/ajax/jQuery/jquery-1.9.1.min.js"></script>
…

<script>
    $(document).ready(function(){
        $("#myTag").click( function() {
            getMessageFromServer();
        });
    });

    function getMessageFromServer()
    {
        ...
    }
</script>

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <h1 id="myTag">Hello World!</h1>
    </body>
</html>
```

No mixing of layout and logic

Quelle: http://www.w3schools.com/jquery

AJAX & jQuery

Quelle: http://www.w3schools.com/jquery

AJAX

$.ajax() method for making Browser-independent XMLHttpRequests.

```
$.ajax({
     url: 'myAjaxServlet',
     data: {myName : 'Grabowski'},
     type: 'GET',
     async: true,
     dataType : 'text'
})
.done(function(answer) {
     alert("From server: " + answer);
})
.fail(function(xhr, status, errorThrown) {
     alert("Sorry, there was a problem!");
})
.always(function(xhr, status) {
     alert("The request is complete!");
})
```

Quelle: https://learn.jquery.com/ajax/jquery-ajax-methods/

JSON (JavaScript Object Notation)

Principal: "Key" : "Value"

```
{
  "Herausgeber": "Xema",
  "Nummer": "1234-5678-9012-3456",
  "Deckung": 2e+6,
  "Waehrung": "EURO",
  "Inhaber":
  {
    "Name": "Mustermann",
    "Vorname": "Max",
    "maennlich": true,
    "Hobbys": [ "Reiten", "Golfen", "Lesen" ],
    "Alter": 42,
    "Kinder": [],
    "Partner": null
  }
}
```

Quelle: https://de.wikipedia.org/wiki/JavaScript_Object_Notation/

JSON and JavaScript

Convert JavaScript-Object to JSON-String

```
var myObj = { "name":"John", "age":31, "city":"New York" };
var myJSON = JSON.stringify(myObj);
```

Convert JSON-String to JavaScript-Object

```
var myJSON = '{ "name":"John", "age":31, "city":"New York" }';
var myObj = JSON.parse(myJSON);
```

AJAX and JSON

$.ajax() method for making Browser-independent XMLHttpRequests.

```
$.ajax({
     url: 'myAjaxJSONServlet',
     data: {myName : 'Grabowski'},
     type: 'GET',
     async: true,
     dataType : 'json'
})
.done(function(answer) {
     alert("From server: " + answer.title);
})
.fail(function(xhr, status, errorThrown) {
     alert("Sorry, there was a problem!");
})
.always(function(xhr, status) {
     alert("The request is complete!");
})
```

Automatic conversion to JavaScript-Object

Quelle: https://learn.jquery.com/ajax/jquery-ajax-methods/

AJAX and “same-origin policy”

AJAX-request must have same origin as first web-page request.

```
$.ajax({
     url: ' https://de.wikipedia.org/…',
     data: {titles: 'Grabowski'},
     type: 'GET',
     async: true,
     dataType : 'json'
})
.done(function(answer) {
     alert("From server: " + answer.query);
})
.fail(function(xhr, status, errorThrown) {
     alert("Sorry, there was a problem!");
})
.always(function(xhr, status) {
     alert("The request is complete!");
})
```

Forbidden by
Same-Origin-Policy

AJAX and JSONP

Hack: JSONP (JSON with Padding)

1. Client dynamically *"injects"* a script element
```
<script type="application/javascript"
  src="http://www.wikipedia.de/api?callback=parseResponse">
</script>
```

2. Server generates a script containing the data and sends it back
```
parseResponse({"Name": "Foo", "Id": 1234, "Rank": 7});
```

3. Client has generated a JavaScript function to access the JSON-Elements
```
parseResponse = function (result)
  { var obj = JSON.parse(result);
    alert(obj.Name + ' ' + obj.ID + ' ' + obj.Rank);
  };
```

jQuery and JSONP

```
$.ajax({
   url: "http://de.wikipedia.org/api/…",

   // The name of the callback parameter, as specified by server
   jsonp: "callback",

   // Tell jQuery we're expecting JSONP
   dataType: "jsonp",

   // Tell server what we want and that we want JSON
   data: { titles: "Oli”, format: "JSON”},

   // Work with the response
   success: function( response ) {
      console.log( response ); // server response
      obj1 = response.query;
      obj2 = response. …   ;
      ...
   }
});
```

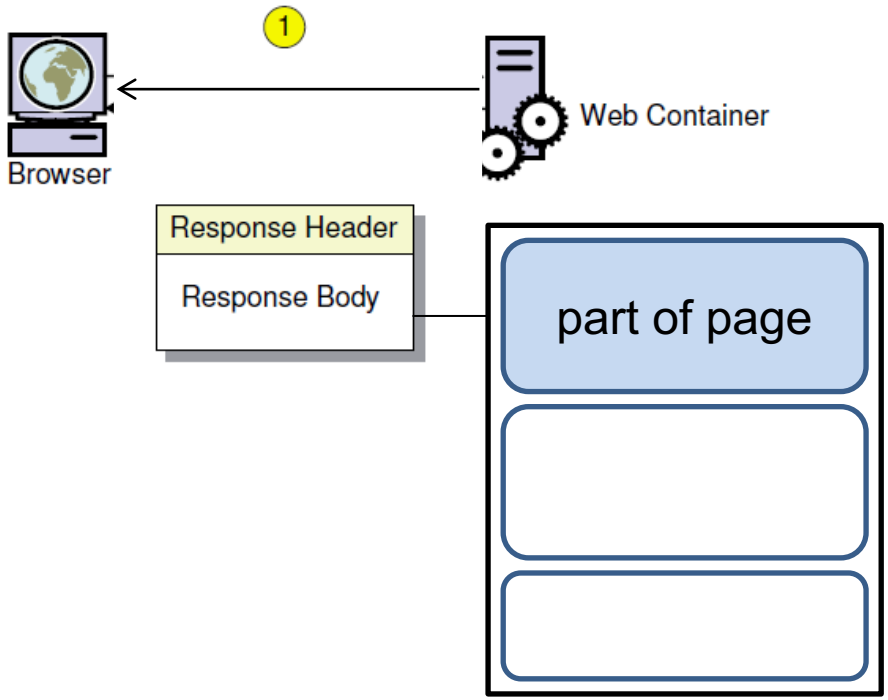
https://learn.jquery.com/ajax/working-with-jsonp/

Reverse AJAX

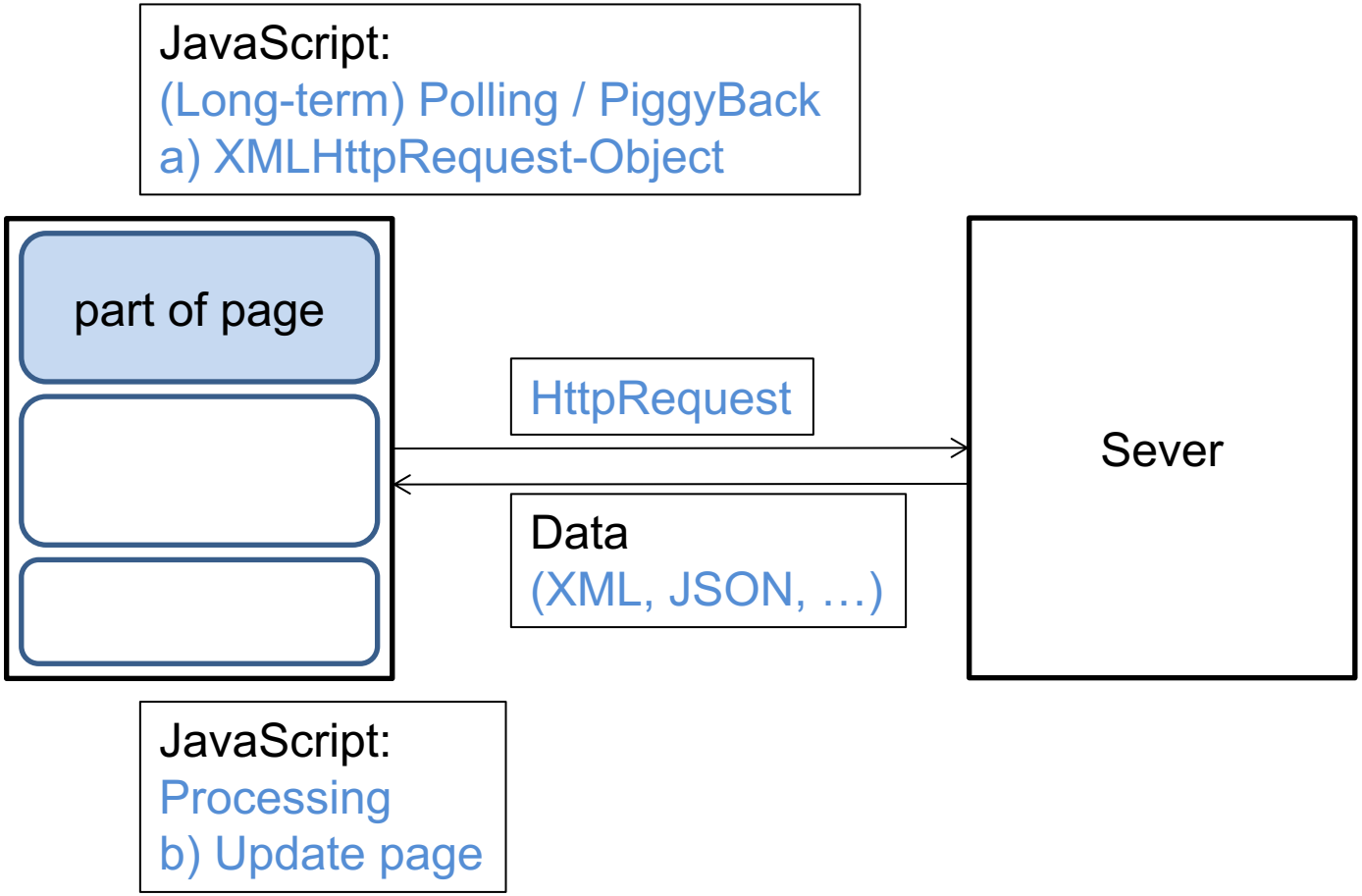
Quelle: http://www.w3schools.com/jquery

Modul 6
Motivation

Server wants to push some data.



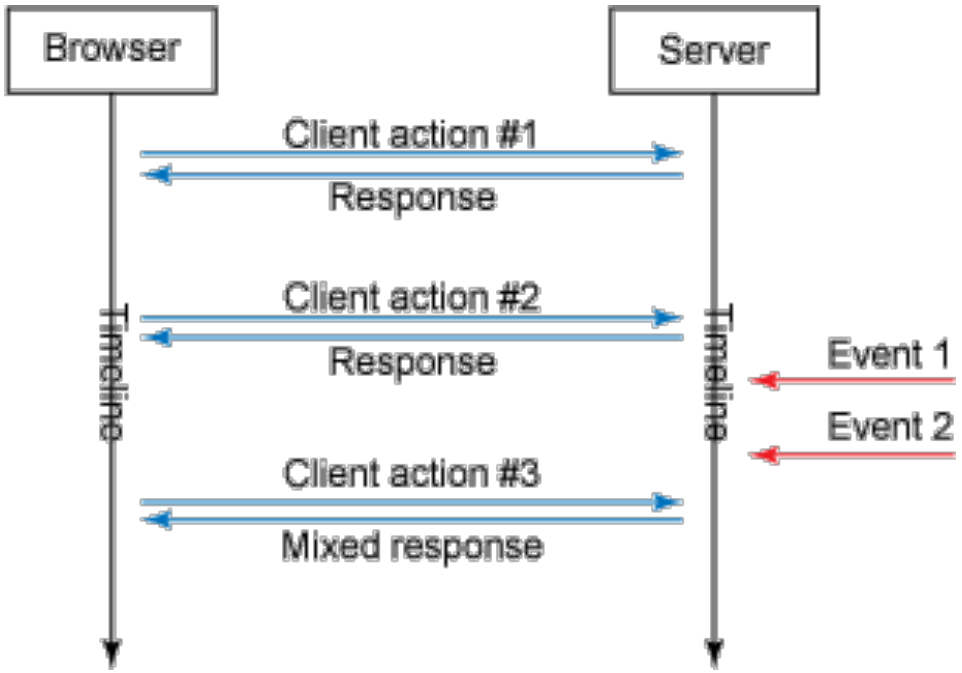
Modul 6
Solution



Modul 6
Reverse Ajax: Piggyback

Piggyback

With the piggyback option, the server, having an update to send, waits for the next time the browser makes a connection and then sends it's update along with the response that the browser was expecting.

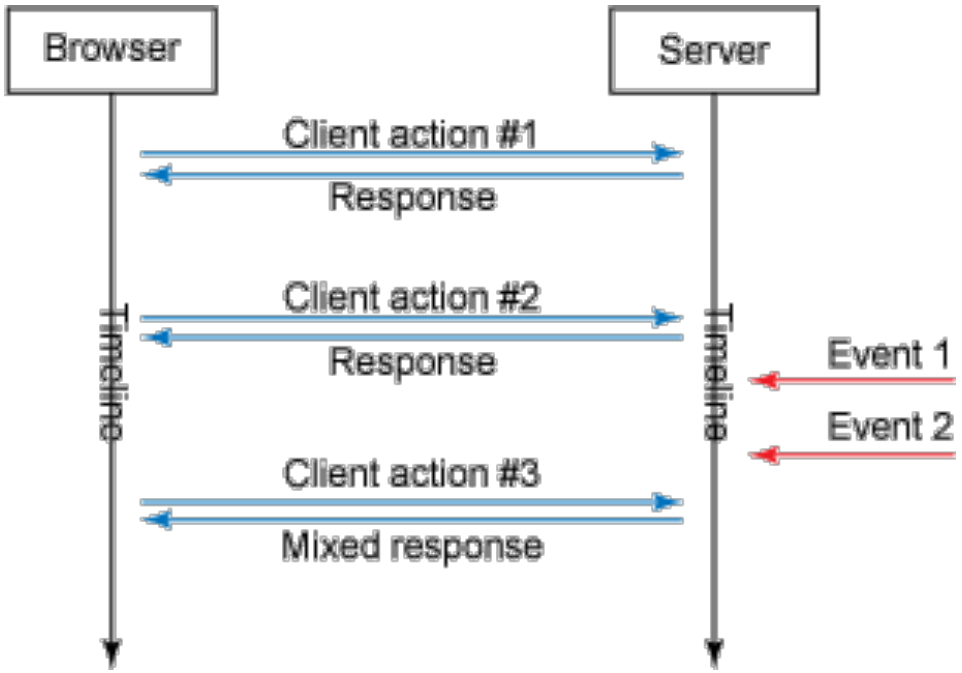


Quelle: http://www.ibm.com/developerworks/library/wa-reverseajax1/

Modul 6
Reverse Ajax: Piggyback

Polling

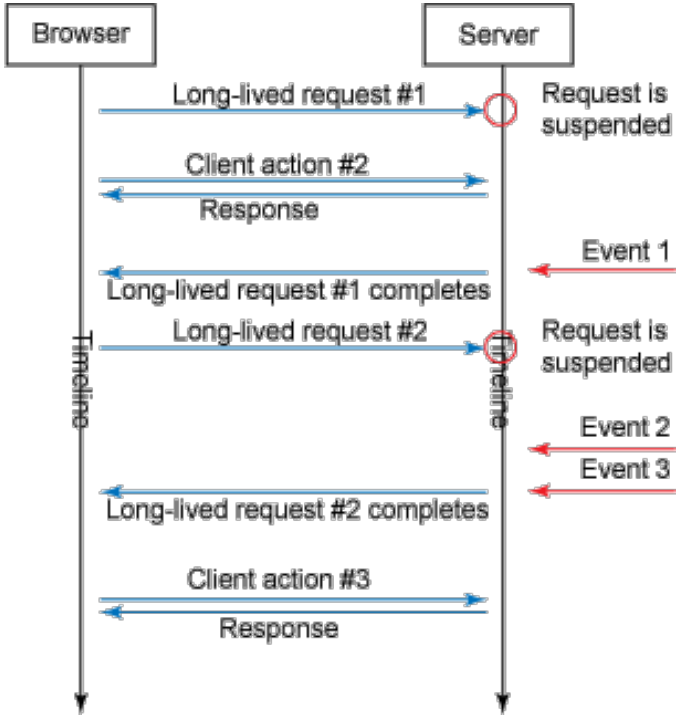
The browser makes a request of the server at regular and frequent intervals, to see if there has been an update to the page.



Quelle: http://www.ibm.com/developerworks/library/wa-reverseajax1/

Comet

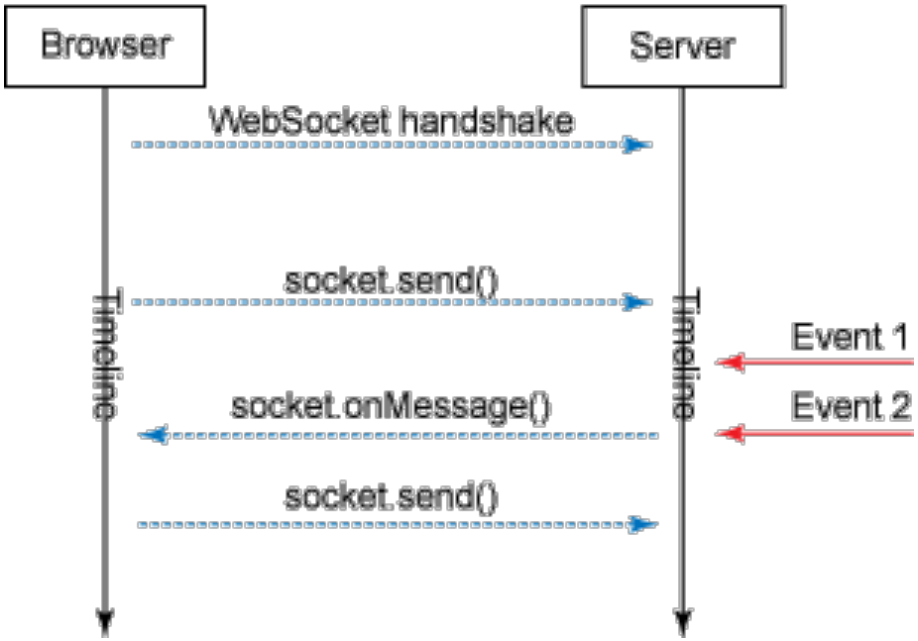
Comet is a web application model where a request is sent to the server and kept alive for a long time, until a time-out or a server event occurs. When the request is completed, another long-lived Ajax request is sent to wait for other server events.



Quelle: http://www.ibm.com/developerworks/library/wa-reverseajax1/

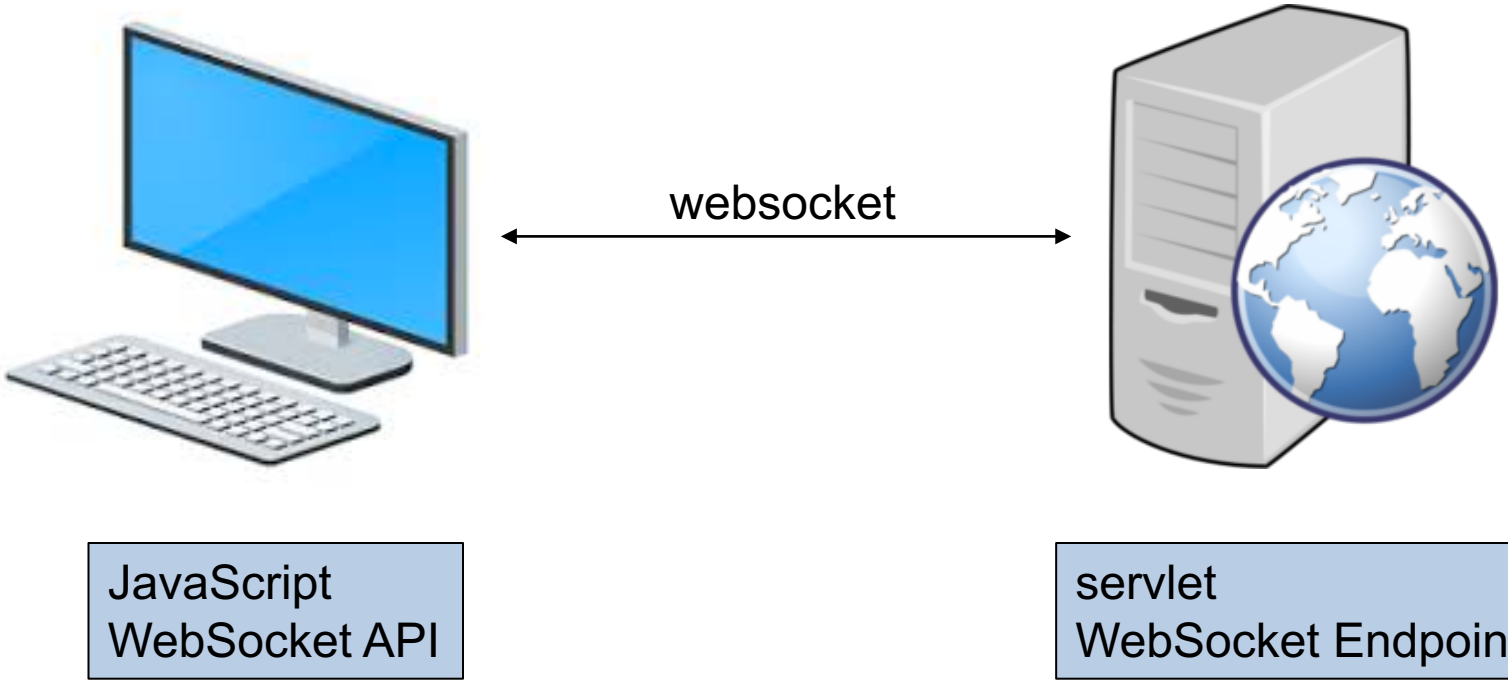
WebSockets

WebSockets, which emerged in HTML5, enables bi-directional, full-duplex communication channels. The connection is opened through an HTTP request in JavaScript. A WebSocket URL is started by typing **`ws:// or wss://`** (on SSL).



Quelle: http://www.ibm.com/developerworks/web/library/wa-reverseajax2/index.html/

WebSockets


websocket
JavaScript
WebSocket API
servlet
WebSocket Endpoint

JavaScript WebSocket APIs

```
<script>
    var wsUri = "ws://localhost:8080/M06_Reverse_AJAX/endpoint";
    var websocket = new WebSocket(wsUri);

    //Empfangen von Daten
    websocket.onmessage = function (e) {
        console.log(e.data);
    };

    websocket.onerror = function (e) {
       console.log(e);
    };

    websocket.onopen = function (e) {
       console.log(e);
    };
</script>
```

https://developer.mozilla.org/de/docs/Web/API/WebSocket

JavaScript WebSocket APIs

```
<script>

    var wsUri = "ws://localhost:8080/M06_Reverse_AJAX/endpoint";
    var websocket = new WebSocket(wsUri);

    //Senden von Daten
    websocket.send('My Message');

</script>
```

https://developer.mozilla.org/de/docs/Web/API/WebSocket

WebSockets: Annotated Endpoints

```
@ServerEndpoint("/mySocket")
public class MyEndpoint {
   @OnMessage
    public void onMessage(Session session, String msg) {
        try {
            console.log("Received from: " + session.getId());
            console.log("Received: " + msg);

        } catch (IOException e) { ... }
   }
}
```

Note: As opposed to servlets, WebSocket endpoints ***are instantiated multiple times***. The container creates an instance of an endpoint per connection to its deployment URI. Each instance is associated with one and only one connection.

https://docs.oracle.com/javaee/7/tutorial/websocket004.htm

WebSockets: Annotated Endpoints

```
@ServerEndpoint("/mySocket")
public class MyEndpoint {

   @OnOpen
   public void onOpen (Session session) throws InterruptedException {
         System.out.println("Serverside Open: " + session.getId());
   }

   @OnClose
   public void onClose (Session session) {
       System.out.println("Serverside Close: " + session.getId());
   }

   @OnError
   public void error(Session session, Throwable error) {
       System.out.println("Serverside Close: " + session.getId());
   }
}
```

https://docs.oracle.com/javaee/7/tutorial/websocket004.htm

WebSockets: Receiving Messages

```
@ServerEndpoint("/mySocket")
public class MyEndpoint {

   @OnMessage
   public void onMessage(Session session, String msg) {
      try {
         console.log("Received: " + msg);
       } catch (IOException e) { ... }
   }

}
```

https://docs.oracle.com/javaee/7/tutorial/websocket004.htm

WebSockets: Sending Messages

```
@ServerEndpoint("/mySocket")
public class MyEndpoint {

   @OnMessage
   public void onMessage(Session session, String msg) {
      try {
         console.log("Received: " + msg);

         session.getBasicRemote().sendText("OK");

       } catch (IOException e) { ... }
   }
}
```

For sending data to a client, the associated session-object is required.

Solution: Store all session-objects in a **`Set`** or **`ArrayList`**.

https://docs.oracle.com/javaee/7/tutorial/websocket004.htm

WebSockets: Sending Messages

```
@ServerEndpoint("/mySocket")
public class MyEndpoint {

   // Store all sessions
   private static Set<Session> peers =
                     Collections.synchronizedSet(new HashSet<Session>());

   @OnOpen
   public void onOpen (Session session) throws InterruptedException {
       peers.add(session);
   }

   @OnClose
   public void onClose (Session session){
      peers.remove(session);
   }
}
```

https://netbeans.org/kb/docs/javaee/maven-websocketapi.html

WebSockets: Sending Messages to All Peers Connected to an Endpoint

```
@ServerEndpoint("/echoall")
public class EchoAllEndpoint {

@OnMessage
   public void onMessage(Session session, String msg) {
      try {
         for (Session sess : session.getOpenSessions()) {
            if (sess.isOpen())
               sess.getBasicRemote().sendText(msg);
         }
      } catch (IOException e) { ... }
   }

}
```

`session.getOpenSessions()` returns a copy of the Set of **all the open web socket sessions** that represent connections to the same endpoint to which this session represents a connection.

https://netbeans.org/kb/docs/javaee/maven-websocketapi.html