

# COMET

“behind closed doors...”

# This presentation is not about

- JavaScript frameworks
- Demos with cool JavaScript effects
- Dojo and its creators, Cometd
- Ajax in general
- Cool things you can do with your website

# This presentation is

- A brief introduction to a great and important “fulhack” that is implemented in web-containers to provide better support for asynchronous communication between the server and the browser....

# web “architectures”

- Page-by-page (long time ago)
  - Traditional Java IO model, thread per connection
- Ajax
  - Ask for data and then only modify the page
- Comet
  - Keep the conversation a live, no matter what (like a bad date)

# “Streaming”

- Server push, reverse ajax
  - Checking/asking the server for new data
  - Not “real” asynchronous communication
  - No need for servers to be modified
  - Ajax stuff

# Sync / Async

- Simple example
  - Our HttpClient library is able to handle a type of async communication

# synchronous

```
HttpClient pseudoHttp = New HttpClient();
```

```
Request req = new HttpRequest("www.fancysitethatresponds.com");
```

```
Response res = req.call(req); //block
```

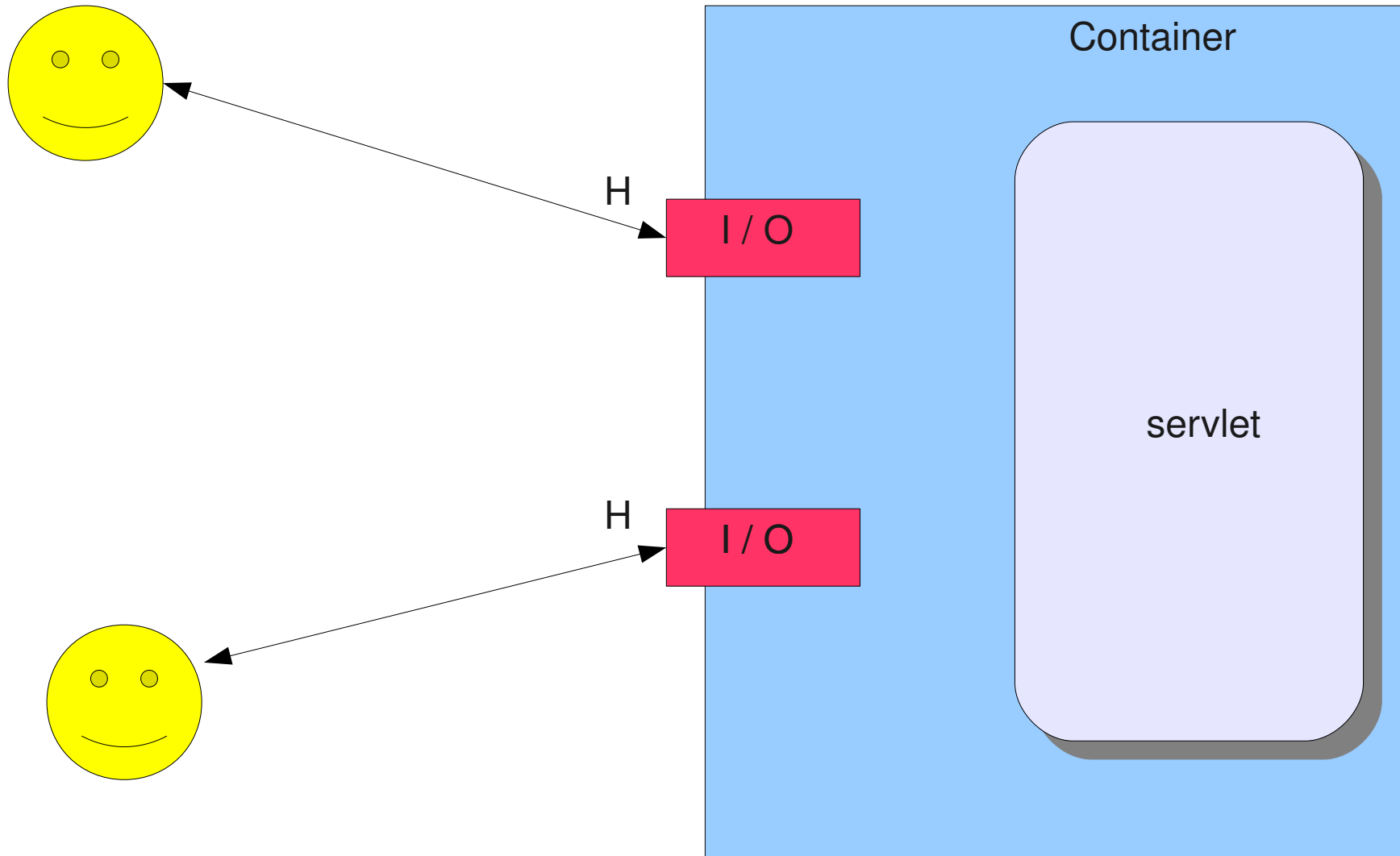
*Meanwhile block everything.*

*No handlers needed here*

# synchronous

- Basic old fashion way of getting a page...
  - the caller thread is suspended until the server response returns or a timeout is exceeded
- Number of threads = number of outstanding requests

# synchronous



# synchronous

- Thread per connection Java Application Server
  - blocking further actions
- Connections
  - May be terminated
- Limitations
  - threads
  - memory
  - sockets

# asynchronous

- Number of requests are “not” restricted
- No suspension of threads until response
- Often with the help of Java NIO

# asynchronous

```
HttpClient pseudoClient = new HttpClient();
```

```
MyResponseHandler () {
```

```
    onResponse() {
```

```
        DO STUFF!
```

```
    }
```

```
}
```

```
-----
```

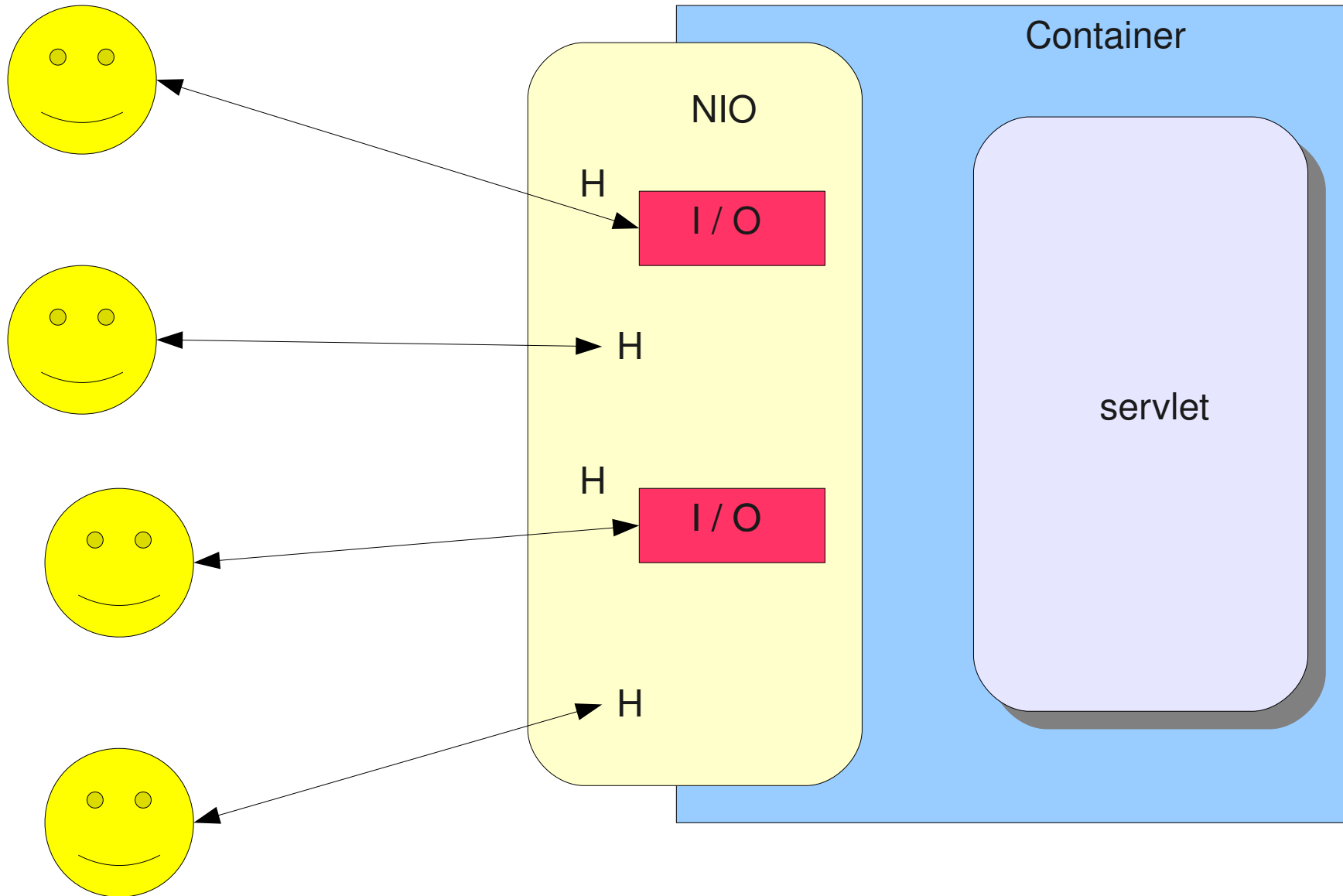
```
Request req = new Request("www.google.com");
```

```
makeRequest(req, aResponseHandler);
```

# asynchronous

- Thread per request
- Idle connections
  - As they are async, they can be able to idle and be maintainable
- Limitations
  - Sockets
  - Latency

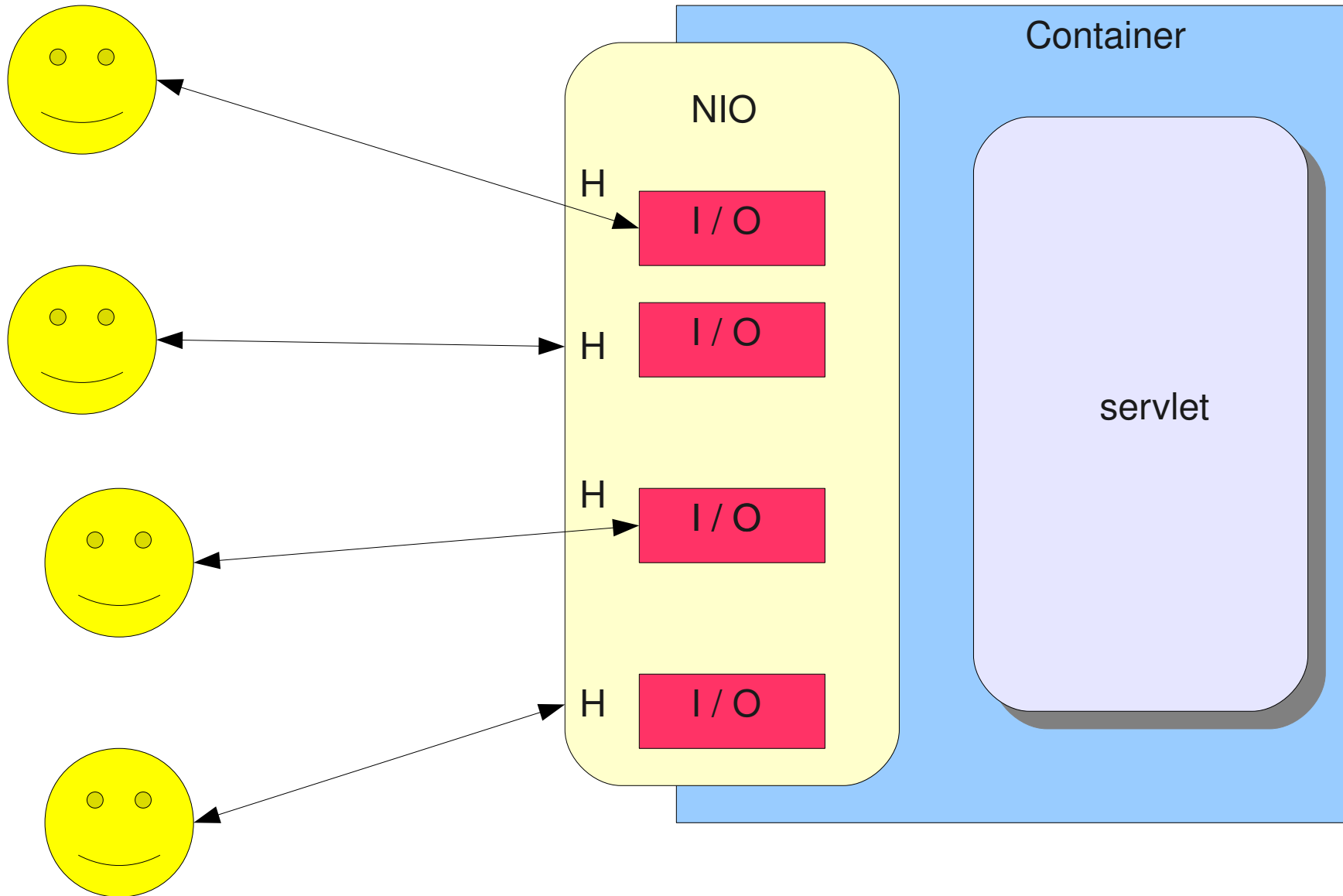
# asynchronous



# Comet

- Long lasting connection
- Long lasting requests
- We just spoke about the “thread per request” “strength”
- This means thread per request AND(+) client

# Comet (no magic)



# Any problems?

- Thread per request AND client
  - Running out of threads memory etc...
- Servlet 2.5, running plain
  - No support for non-blocking data streaming
  - No support for async

# Comet strategies

- Long poll
  - Connection open
    - Until timeout or event occurs
    - New call opens up after response
    - Takes up socket buffers

# Comet strategies

## Streamed response

- Constant connection, async
  - Non blocking implementatio needed
  - Data is constantly pushed

# Simple client example

```
<html>
```

```
<head>
```

```
<script language="Javascript">
```

```
function printTime(time) {
```

```
    document.getElementById('text').innerHTML = time;
```

```
}
```

```
</script>
```

```
</head>
```

# Simple client example

```
<body>
```

```
  <iframe id="iframe" name="iframe" width ="0" height ="0" border="0">
```

```
  </iframe>
```

```
  <a href ="time" target="iframe"/>start timer</a>
```

```
  <p id="text">0</p>
```

```
</body>
```

```
</html>
```

# The same with server

```
public void run() {  
    try {  
        String script = "<script>\r\n" +  
            " parent.printTime(\"\" + new Date().toString() + "\");\r\n" +  
            "</script>";  
        outChannel.write(script);  
    } catch (IOException ioe) {  
        cancel();  
        try {  
            outChannel.close();  
        } catch (IOException ignore) {}  
    }  
}  
};
```

# Behind closed doors

HTTP/1.1 200 OK

Content-Type: text/html; charset=iso-8859-1

Transfer-Encoding: chunked

Server: xSocket-http/2.0-alpha-3

**<script>**

**parent.printTime("Sun Feb 10 10:05:20 CET 2008");**

**</script>**

**<script>**

**parent.printTime("Sun Feb 10 10:05:22 CET 2008");**

**</script>**

**<script>**

**parent.printTime("Sun Feb 10 10:05:23 CET 2008");**

**</script>**

# Comet & Tomcat

- CometProcessor (extends javax.servlet.Servlet)
  - Service method is never called
  - Events are triggered
    - BEGIN
    - ERROR
    - READ

# Comet & Tomcat

```
public void init() throws ServletException {  
    Thread timeSenderThread = new Thread(new TimeSender());  
    timeSenderThread.setDaemon(true);  
    timeSenderThread.start();  
}
```

```
HttpServletRequest req = event.getHttpServletRequest();
```

```
HttpServletResponse resp = event.getHttpServletResponse();
```

# Comet & Tomcat

```
if (event.getEventType() == CometEvent.EventType.BEGIN) {  
    synchronized(connections) {  
        connections.add(resp);  
    }  
}
```

# Comet & Tomcat

```
class TimeSender
```

```
public void run() {
```

```
    while (true) {
```

```
        synchronized (connections) {
```

```
            for (HttpServletResponse response : connections) {
```

```
                PrintWriter bodyStream = response.getWriter();
```

```
                String script = "<script>\r\n" +
```

```
                    "    parent.printTime(\"\" + new Date() + "\");\r\n" +
```

```
                    "</script>";
```

```
                bodyStream.write(script);
```

```
                bodyStream.flush();
```

```
            }
```

```
        }
```

# Comet & Jetty

- Supports a “third model” of our Comet “vision”
  - Statement “thread per client and request” is no longer true
- Continuation
  - Release a thread when suspended (to the container)
  - Save request for future processing

# Comet & Jetty

- `RetryRequest` exception
  - Caught by the container
  - Release thread
- `StoreControlState`
  - stack trace, variables, and program counter
  - Reconstructed when “back”

# Comet & Jetty

```
Continuation cc = ContinuationSupport.getContinuation(req, null);
```

```
while (true) {  
    cc.suspend(1000); // suspend the response (free)  
  
    String script = "<script>\r\n" +  
        "    parent.printTime(\"\" + new Date() + "\");\r\n" +  
        "</script>";  
  
    resp.getWriter().println(script);  
  
    resp.getWriter().flush();  
}
```

# Why this is cool



# Comet & World

- DWR
- Cometd
  - servlet implementation of the Bayeux protocol of cometd from the Dojo Foundation
- Grizzly
- Jetty-continuation & Tomcat
- Servlet 3.0

COMET

DEMO

Comet

Q&A