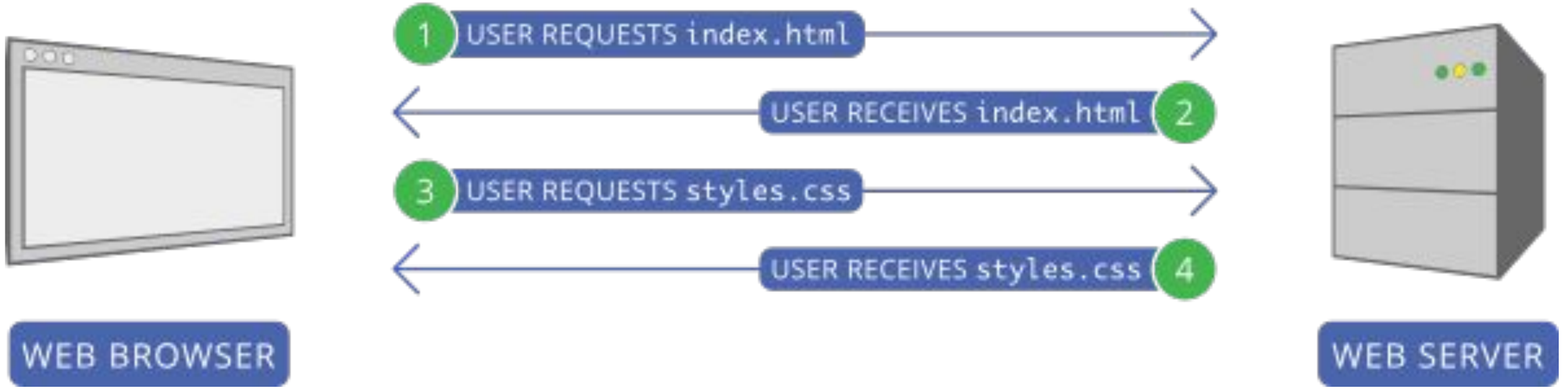
# Server Push Technology

Malintha Adikari

# TYPICAL WEB SERVER COMMUNICATION



**Problems with this mechanism:**

- Forces the user to wait for the browser to discover and retrieve critical assets until after an HTML document has been downloaded.
- Delays rendering and increases load times.
- No way to get state changes of the server without client initialized request

  Ex: Chat applications

  Stock Market updates

# What is the solution?

**Polling :(**

**Any other solutions**

# Server Push Technology

- Is a style of Internet-based communication where the request is initiated by the central server
- Push allows a web server to send resources to a web browser before the browser gets to request them

# Reverse AJAX

- Gives the ability to asynchronously send data from a web-server to a browser.
- The goal is to let the server push information to the client.
- Ajax requests are stateless by default, and can only be opened from the client to the server.

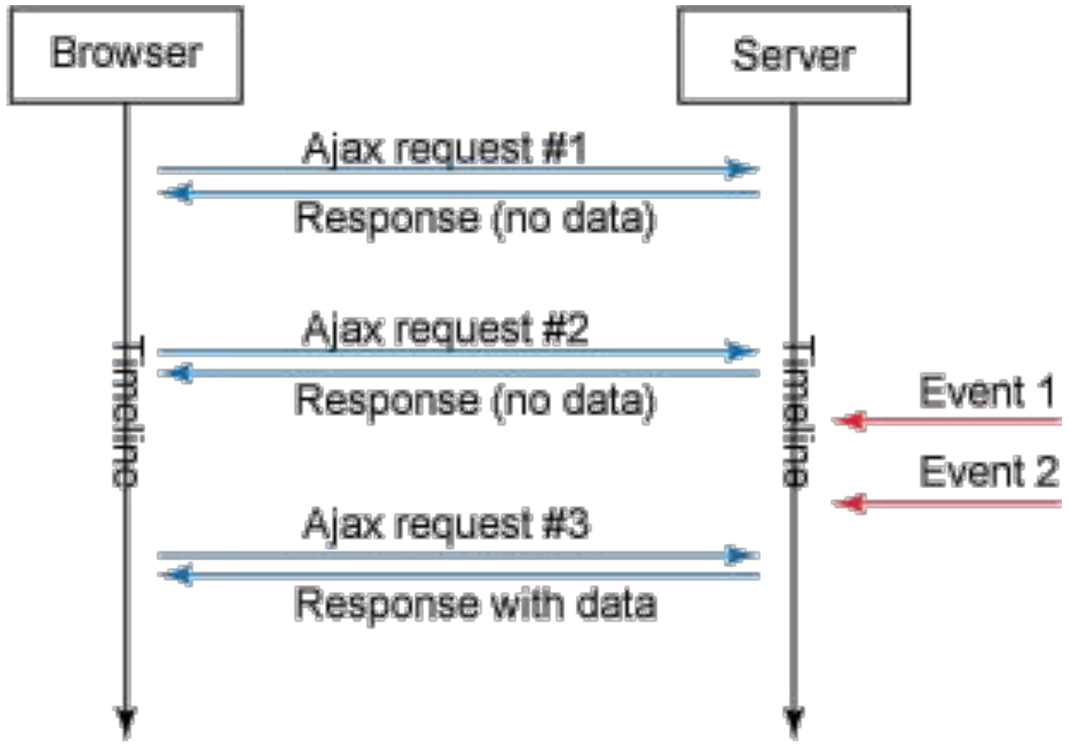
# HTTP Polling

]
\

- Polling involves issuing a request from the client to the server to ask for some data.
- Polling interval (time between requests) must be as low as possible to get the server events as soon as possible
- **Drawback**: if this interval is reduced, the client browser is going to issue many more requests, and will consume bandwidth and processing resources for nothing.

# HTTP Polling

Ex: The client must wait for the next polling to get the two events received by the server.




Reverse Ajax with HTTP polling

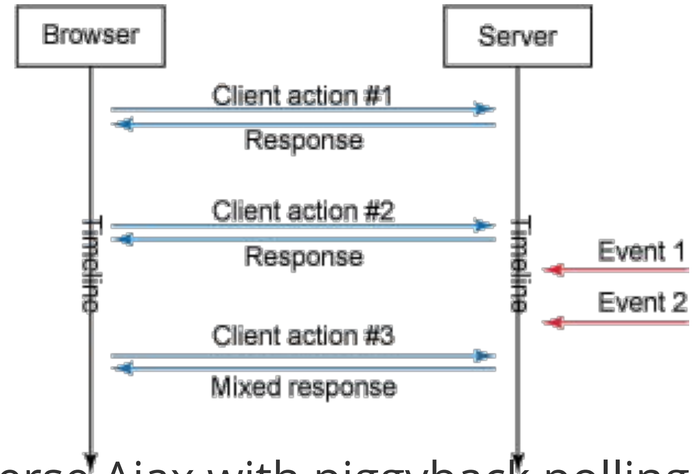
# HTTP Polling

- **Advantages**:
  - Easy to implement
  - Does not require any special features on the server side.
  - Works in all browsers.

- **Disadvantage**:
  - Rarely employed because it does not scale.
  - Ex: case of 100 clients each issuing polling requests for 2 seconds, where 30% of the requests returned no data.

# Piggyback Polling

- Much more clever method than polling since it tends to remove all non-needed requests (those returning no data).
- There is no interval; requests are sent when the client needs to send a request to the server.
- The difference lies in the response, which is split into two parts: the response for the requested data and the server events, if any occurred.




Reverse Ajax with piggyback polling

# Piggyback Polling

- **Advantages**:
  - With no requests returning no data, since the client controls when it sends requests, you have less resource consumption.
  - Works in all browsers and does not require special features on the server side.

- **Disadvantages**:
  - No clue when the events accumulated on the server side will be delivered to the client because it requires a client action to request them.

# Comet

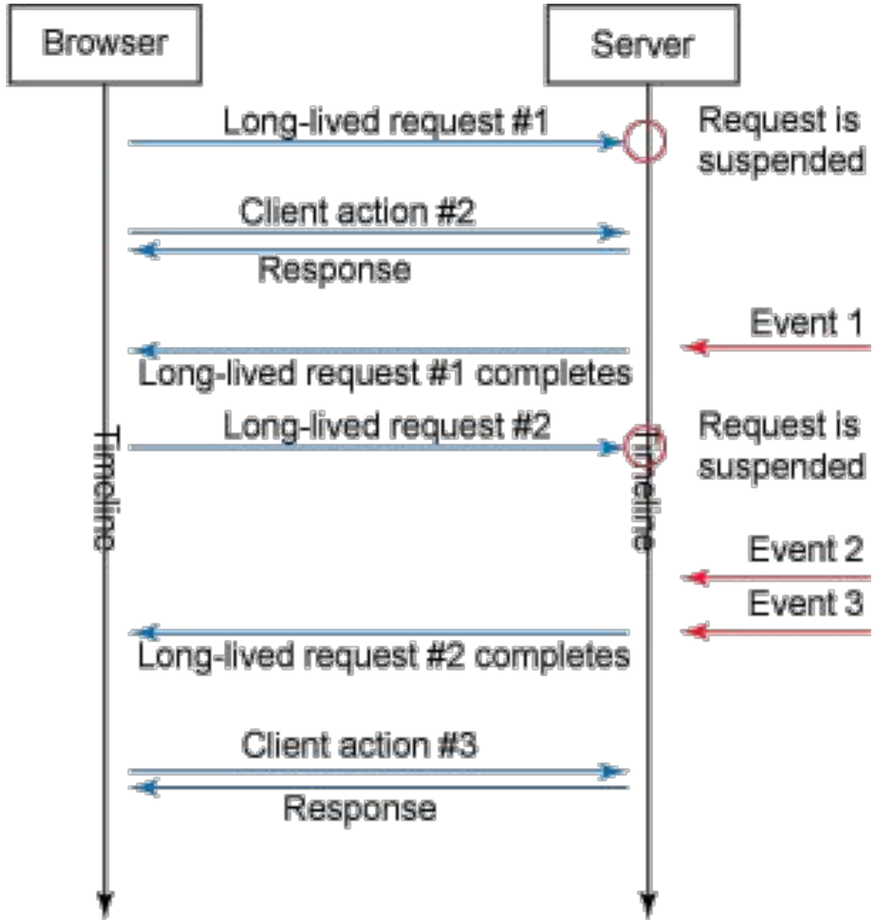
- Comet is an umbrella term, encompassing multiple techniques for achieving server push interaction.
- Reverse Ajax with polling or piggyback is very limited: it does not scale and does not provide low-latency communication.
- Comet is a web application model where a request is sent to the server and 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.
- With Comet, web servers can send the data to the client without having to explicitly request it.

# Comet

**Advantage**: Each client always has a communication link open to the server.

- Server can push events on the clients by immediately committing the responses when they arrive, or it can even accumulate and send bursts.

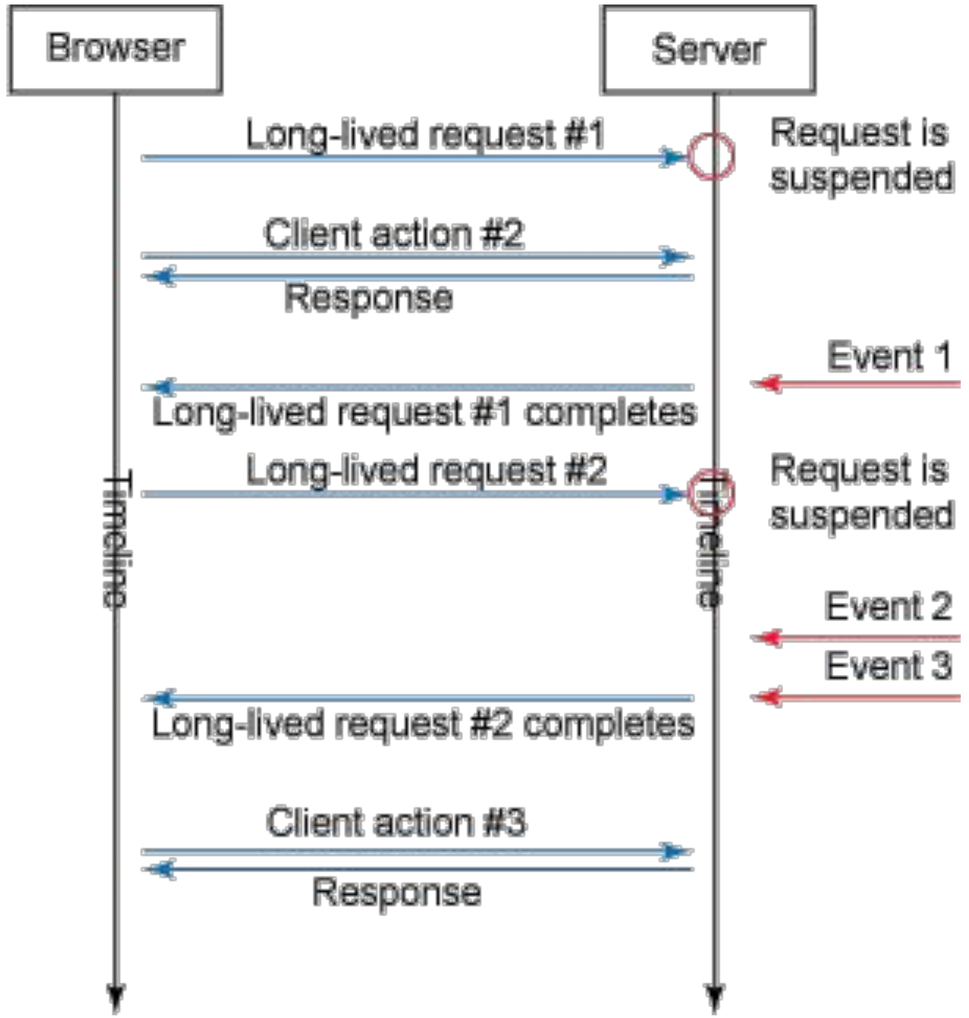
- Because a request is kept open for a long time, special features are required on the server side to handle all of these long-lived requests.



# Comet

Implementations of Comet can be separated into two types:

- streaming
- long polling.



# Comet using HTTP Streaming

- One persistent connection is opened.
- There will only be a long-lived request since each event arriving on the server side is sent through the same connection.
- It requires on the client side a way to separate the different responses coming through the same connection.
- Two common techniques for streaming
  - Forever Iframes (hidden IFrames)
  - Multi-part feature of the XMLHttpRequest object used to create Ajax requests in JavaScript.

# Forever IFrames

- Technique involves a hidden Iframe tag put in the page with its **src** attribute pointing to the servlet path returning server events.
- Each time an event is received, the servlet writes and flushes a new script tag with the JavaScript code inside.
- The iframe content will be appended with this script tag that will get executed.

- **Advantages**: Simple to implement, works in all browsers supporting iframes.
- **Disadvantages**: No way to implement reliable error handling or to track the state of the connection, because all connection and data are handled by the browser through HTML tags.

# Multi-part XMLHTTPRequest

- Use the multi-part flag supported by some browsers on the XMLHttpRequest object.
- An Ajax request is sent and kept open on the server side.
- Each time an event comes, a multi-part response is written through the same connection.

**JavaScript code to set up a multi-part streaming request**

```
var xhr = $.ajaxSettings.xhr();
xhr.multipart = true;
xhr.open('GET', 'ajax', true);
xhr.onreadystatechange = function() {
    if (xhr.readyState == 4) {
        processEvents($.parseJSON(xhr.responseText));
    }
};
xhr.send(null);
```

# Multi-part XMLHTTPRequest

On the server side, First set up the multi-part request, and then suspend the connection.

Suspending an HTTP streaming request in a servlet

```
protected void doGet(HttpServletRequest req, HttpServletResponse resp)
      throws ServletException, IOException {
   // start the suspension of the request
   AsyncContext asyncContext = req.startAsync();
   asyncContext.setTimeout(0);

   // send the multipart separator back to the client
   resp.setContentType("multipart/x-mixed-replace;boundary=\""
      + boundary + "\"");
   resp.setHeader("Connection", "keep-alive");
   resp.getOutputStream().print("--" + boundary);
   resp.flushBuffer();

   // put the async context in a list for future usage
   asyncContexts.offer(asyncContext);
}
```

# Multi-part XMLHTTPRequest

Each time an event occurs, iterate over all suspended connections and write the data to them

Send events to a suspended multi-part request using Servlet

```
for (AsyncContext asyncContext : asyncContexts) {
    HttpServletResponse peer = (HttpServletResponse)
        asyncContext.getResponse();
    peer.getOutputStream().println("Content-Type: application/json");
    peer.getOutputStream().println();
    peer.getOutputStream().println(new JSONArray()
        .put("At " + new Date()).toString());
    peer.getOutputStream().println("--" + boundary);
    peer.flushBuffer();
}
```

# Multi-part XMLHTTPRequest

- **Advantages**:
  - Only one persistent connection is opened.
  - This is the Comet technique that saves the most bandwidth usage.

- **Disadvantages**:
  - The multi-part flag is not supported by all browsers.
  - Some widely used libraries, such as CometD in Java, reported issues in buffering. For example, chunks of data (multi-parts) may be buffered and sent only when the connection is completed or the buffer is full, which can create higher latency than expected.

# Comet Using HTTP Long-Polling

- The long polling mode involves techniques that open a connection.
- The connection is kept open by the server, and, as soon as an event occurs, the response is committed and the connection is closed.
- Then, a new long-polling connection is reopened immediately by the client waiting for new events to arrive.
- Can implement HTTP long polling by using script tags or a mere XMLHttpRequest object.

# Comet Using HTTP Long-Polling

**Script tags**

- The goal is to append a script tag in your page to get the script executed.
- The server will suspend the connection until an event occurs, send the script content back to the browser, and then reopen another script tag to get the next events.
- **Advantages**: Very easy to implement and works across domains (by default, XMLHttpRequest does not allow requests on other domains or sub-domains).
- **Disadvantages**: Similar to the iframe technique, error handling is missing, and can't have a state or the ability to interrupt a connection.

# XMLHTTPRequest Long-polling

- Open an Ajax request to the server and wait for the response.
- The server requires specific features on the server side to allow the request to be suspended.
- As soon as an event occurs, the server sends back the response in the suspended request and closes it, exactly like you close the output stream of a servlet response.
- The client then consumes the response and opens a new long-lived Ajax request to the server:

```
function long_polling() {
    $.getJSON('ajax', function(events) {
        processEvents(events);
        long_polling();
    });
}
long_polling();
```

# XMLHTTPRequest Long-polling

Suspending a long polling Ajax request in server

```
protected void doGet(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {
    AsyncContext asyncContext = req.startAsync();
    asyncContext.setTimeout(0);
    asyncContexts.offer(asyncContext);
}
```

# XMLHTTPRequest Long-polling

When an event is received, simply take all of the suspended requests and complete them

```
while (!asyncContexts.isEmpty()) {
    AsyncContext asyncContext = asyncContexts.poll();
    HttpServletResponse peer = (HttpServletResponse)
        asyncContext.getResponse();
    peer.getWriter().write(
        new JSONArray().put("At " + new Date()).toString());
    peer.setStatus(HttpServletResponse.SC_OK);
    peer.setContentType("application/json");
    asyncContext.complete();
 }
```

# XMLHTTP Request Long-polling

- **Advantages**:
  - Easy to implement on the client side with a good error-handling system and timeout management.
  - Reliable technique also allows a round-trip between connections on the server side, since connections are not persistent .
  - Works on all browsers

- **Disadvantages**:
  - Still relies on a stateless HTTP connection, which requires special features on the server side to be able to temporarily suspend it.

# Thanks!

Any questions?