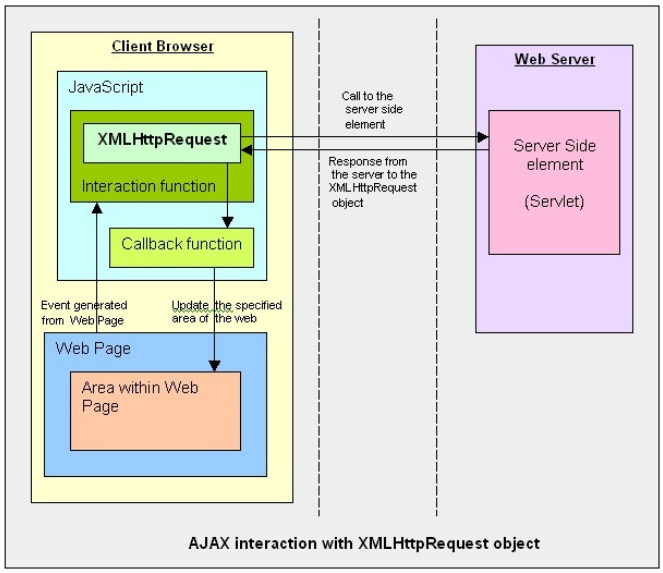
1 Basics

XMLHttpRequest



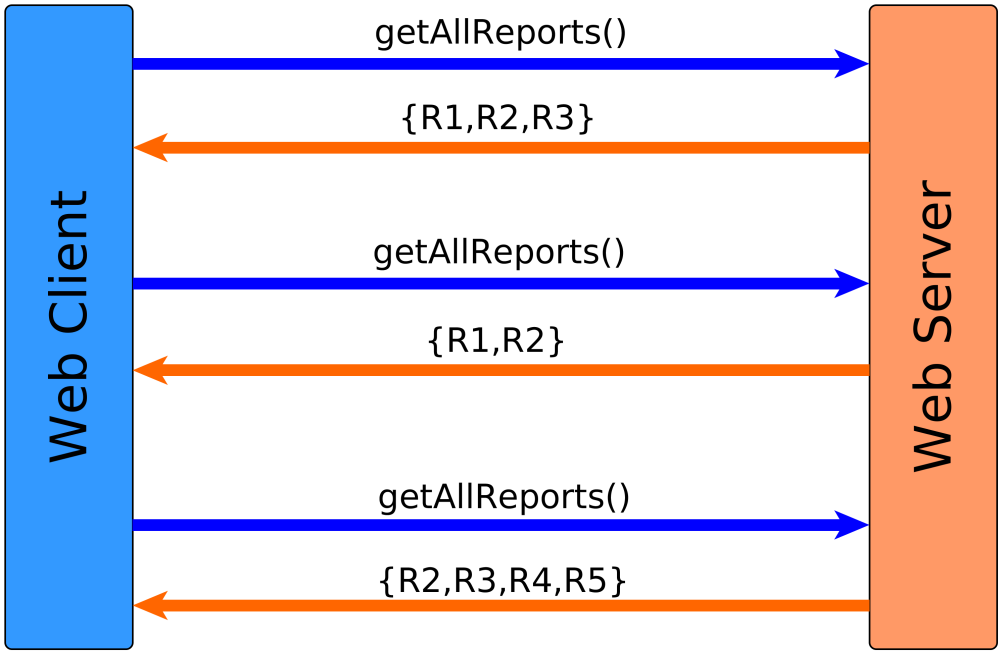
taken from `https://devcentral.f5.com/articles/social-media-abcs-x-is-for-xmlhttp`

The Story so Far

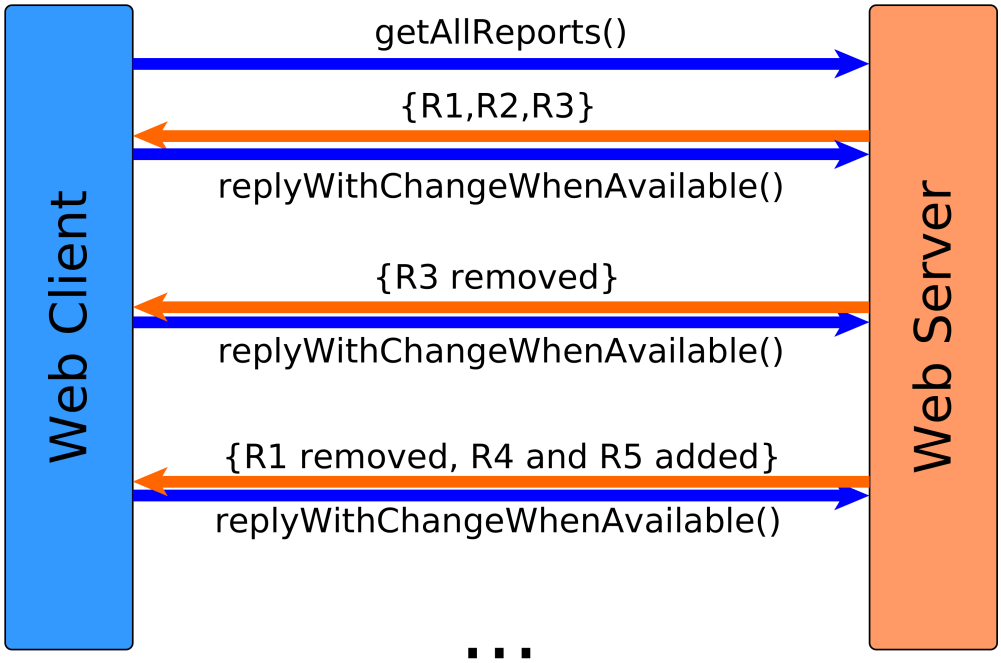
- we have learned technologies to create an application on server-side as well as client-side.
- to communicate we use exclusively the HTTP(S) protocol, typically throught REST.

Problem
What to do when new data on server appear and the client does not know ?

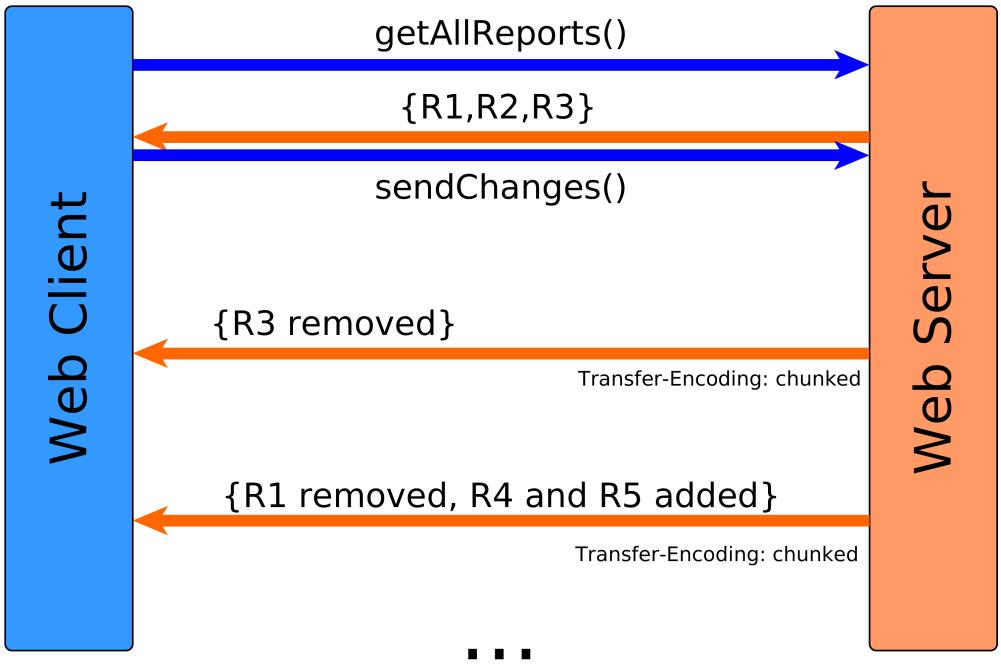
Simple Solution Using HTTP



Better Solution Using HTTP – AJAX push (Long Poll)



Better Solution Using HTTP – AJAX push (Streaming)



... but Still Not Perfect

- the client has to create a new HTTP connection for each communication type (getReports, getUsers, chat)
- HTTP headers have to be sent forth and back for each client side
- the client has to understand/parse low-level HTTP chunks

WebSockets
represent a systematic solution to HTTP client-server pecularities and provides a symmetric model for client-server communication.

Web Socket vs. HTTP

HTTP

- designed for "web pages" not "interactive web applications"
- traditional request-response model
- intensive client-server communication – significant overhead (HTTP headers)

Web Sockets

- bi-directional, full-duplex, real-time,
- low-latency client/server communications on top of TCP/IP
- $\in$ Java EE 7

2 Web Sockets in Java

Web Socket Handshake

```
GET ws://server.org/wsendpoint HTTP/1.1
Host: server.org
Connection: Upgrade
Upgrade: websocket
Origin: http://server.org
Sec-WebSocket-Version: 13
Sec-WebSocket-Key: GhkZiCk+0/91FXIbUuRlVQ==
Sec-WebSocket-Extensions: permessage-deflate; client_max_window_bits
```

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: upgrade
Sec-WebSocket-Accept: jpwu9a/SXDrsoRR26Oa3JUEFchY=
Sec-WebSocket-Extensions: permessage-deflate;client_max_window_bits=15
...
```

Java API for WebSocket (JSR-356)

annotations on POJOs to interact with WebSocket lifecycle events

interfaces to implement to interact with WebSocket lifecycle events

integration with other Java EE technologies – EJB, CDI

JSR-356 Example

```
@ServerEndpoint("/actions")
public class WebSocketServer {

   @OnOpen
   public void open(Session session) { ... }

   @OnClose
   public void close(Session session) { ... }

   @OnError
   public void onError(Throwable error) { ... }

   @OnMessage
   public void handleMessage(String message, Session session) {
      // actual message processing
   }
}
```

JavaScript Side Example

```
var socket = new WebSocket("ws://server.org/wsendpoint");
socket.onmessage = onMessage;

function onMessage(event) {
   var data = JSON.parse(event.data);
   if (data.action === "addMessage") {
      ...
      // actual message processing
   }
   if (data.action === "removeMessage") {
      ...
      // actual message processing
   }
}
```

Other Options

- Spring has wide support through custom annotations - `spring-websocket` module
- ReactJS has `react-websocket` module (listener to WebSocket Events)

Sample Application – Chat

`https://goo.gl/MQMWBf`

Sample Application – Chat Monitoring

- Open Chrome Developer Tools
- Navigate to the web site using Google Chrome
- Open tab “Network” and select the request “actions” (chat)
- Select the subtab “Frames” and you can track the WebSocket communication

References

RFC 6455 - The WebSocket Protocol https://tools.ietf.org/html/rfc6455

JSR 356: Java API for WebSocket https://jcp.org/en/jsr/detail?id=356

Java EE 7: Building Web Applications with WebSocket, JavaScript and HTML5 http://www.oracle.com/webfolder/technetwork/tutorials/obe/java/HomeWebsocket/WebsocketHome.html

Spring Support for WebSocket http://docs.spring.io/spring/docs/current/spring-framework-reference/html/websocket.html