

# WebSockets i WebGL

Nowe protokoły z rodziny WWW

**Jarosław Rzeszółtko**

[sztywny@gmail.com](mailto:sztywny@gmail.com) // [www.stifflog.com](http://www.stifflog.com)

# Kontekst

Dwie przeszkody w przeniesieniu “reszty” oprogramowania do Webu:

- Nieadekwatność schematu zapytanie-odpowiedź w wielu zastosowaniach (opóźnienia)
- Ubóstwo środowiska przeglądarki

Nad rozwiązaniem tych problemów formalnie pracuje W3C:

- HTML5
- WebSockets
- WebGL

A bardzo zainteresowane rozwiązaniem są korporacje:

- Google ChromeOS

# WebSockets

Standardowy protokół, służący wymianianiu danych poprzez **trwałe połączenie**, utrzymywane pomiędzy przeglądarką WWW, a serwerem aplikacji.

Cienka abstrakcja ponad TCP/IP, głównie zapewniające możliwość ograniczenia źródła, z jakiego może być zainicjowane połączenie z serwerem WebSockets.

Eksperyment jeśli chodzi o sposób specyfikacji protokołu.

# Kontrowersje wokół sposobu specyfikacji

Nowy sposób:

Listen on a port for TCP/IP. Upon receiving a connection request, open a connection and send the following bytes back to the client:

```
48 54 54 50 2F 31 2E 31 20 31 30 31 20 57 65 62
20 53 6F 63 6B 65 74 20 50 72 6F 74 6F 63 6F 6C
20 48 61 6E 64 73 68 61 6B 65 0D 0A 55 70 67 72
61 64 65 3A 20 57 65 62 53 6F 63 6B 65 74 0D 0A
43 6F 6E 6E 65 63 74 69 6F 6E 3A 20 55 70 67 72
61 64 65 0D 0A 57 65 62 53 6F 63 6B 65 74 2D 4F
72 69 67 69 6E 3A 20
```

Stary sposób:

```
HTTP/1.1 101 Web Socket Protocol Handshake
Upgrade: WebSocket
Connection: Upgrade
WebSocket-Origin: xyz
```

Hmmm...

## Na przykładzie - handshake

Podstawowy klient:

```
<html>
  <body>
    <script type="text/javascript">
      var ws = new WebSocket("ws://localhost:1234/test");

      ws.onopen = function() {
        alert("Oh hai");
      };

      ws.onmessage = function(event) {
        alert("Received data: " + event.data);
        ws.send("Responding something");
      }

      ws.onclose = function() {
        alert("K thx bai");
      }
    </script>
  </body>
</html>
```

## Na przykładzie - handshake (2)

### Serwer w Ruby:

```
require 'socket'

class ThreadedServer
  def initialize(port, handler)
    tcp_server = TCPServer.new(port)

    loop do
      begin
        connection = tcp_server.accept_nonblock
        handler.new.handle(connection)
      rescue Errno::EAGAIN, Errno::ECONNABORTED,
             Errno::EPROTO, Errno::EINTR
        IO.select([tcp_server])
        retry
      end
    end
  end
end

class WebSocketsHandler
  HANDSHAKE = ["HTTP/1.1 101 Web Socket Protocol Handshake",
              "Upgrade: WebSocket",
              "Connection: Upgrade",
              "WebSocket-Origin: file://",
              "WebSocket-Location: ws://localhost:1234/test",
              "WebSocket-Protocol: OAPP", ""]

  def handle(connection)
    fetch_request(connection)
    HANDSHAKE.each { |line| connection.write(line + "\r\n") }
    connection.write("\x00Hello\xff")
    puts "Received data:" + connection.recvfrom(100).to_s.chop
  end

  private

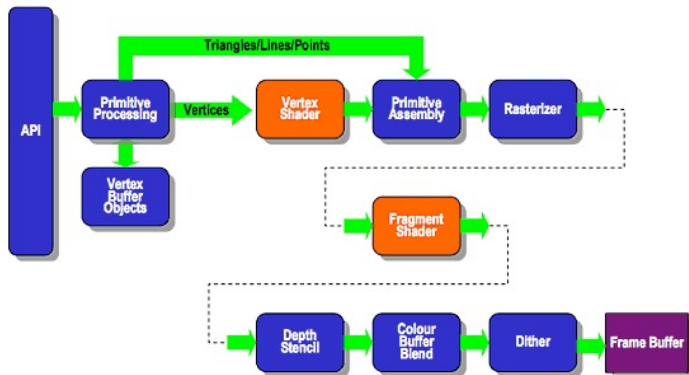
  def fetch_request(connection)
    @request = ""
    while @request.index("\r\n\r\n") == nil do
      @request += connection.gets
    end
  end
end
```

# WebSockets w skrócie

- Cienki wrapper wokół TCP
- Handshake na styl zapytania i odpowiedzi HTTP
- Wymiana dowolnych pakietów zaczynających się bajtem 0x00, kończących się bajtem 0xFF

- Zapoczątkowany przez Mozillę jako Canvas 3D w 2006 roku
- Implementacja OpenGL ES 2.0 w przeglądarce
- Niskopoziome API na elemencie canvas

## ES2.0 Programmable Pipeline



# Rekrutacja

Poszukuje doświadczonych programistów Rails do pracy:

- Atrakcyjne zarobki (5000 - 7000 PLN miesięcznie za pełny etat)
- Praca w nowo powstającym biurze w Warszawie
- Ciekawe projekty dla zagranicznych klientów
- Nowoczesne technologie (Git, BDD, Haml, Sass itp.)

Zainteresowanych proszę o przesłanie CV w języku angielskim na adres:  
**sztywny@gmail.com**