NVDAY

Unity and WebSockets

Adventures in Mobile Games

Josh Glover
Lead Backend Developer

Rock Science 2

What's a WebSocket?

- Bidirectional TCP connection
- Initiated from an HTTP connection by sending Upgrade header

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: x3JJHMbDL1EzLkh9GBhXDw==
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
Origin: http://example.com
```

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: HSmrc0sMlYU=
Sec-WebSocket-Protocol: chat
```

Why WebSockets?

- Server can push messages instead of being polled
- Polling often leads to large percentage of request load being worthless
- Client disconnection can be part of API

Why not WebSockets?

- Servers can’t be purely stateless
- How does a client on server A interact with a client on server B?
- What happens when a server crashes?
- What if a client has a spotty network connection?

Mix and match

- Use WebSockets where remote push is a killer app
- Use plain old REST for everything else
- Gracefully degrade from WebSockets to REST when network connectivity is poor
- HTTP Kit is a perfect fit!

websocket-sharp

- Open source C# WebSockets client
  https://github.com/sta/websocket-sharp/
- Available on the Unity Asset Store
  https://www.assetstore.unity3d.com/en/#!/content/8959
- With Unity 5, drop in your Assets folder and compile away (full .NET 2.0 API only)

Connecting

```
using WebSocketSharp;

var ws = new WebSocket("ws://echo.websocket.org");

// We'll get to this
ws.OnOpen += OnOpenHandler;
ws.OnMessage += OnMessageHandler;
ws.OnClose += OnCloseHandler;
ws.OnError += OnErrorHandler;

ws.ConnectAsync();
```

How do I know when I’m connected?

```
private void OnOpenHandler(object sender, System.EventArgs e) {
    Debug.Log("WebSocket connected!");

    // Here is where you do useful stuff
}
```

Sending a message

```
private void OnSendComplete(bool success) {
    Debug.Log("Message sent successfully? " + success);
}

ws.SendAsync("This WebSockets stuff is a breeze!", OnSendComplete);
```

Receiving a message

```
private void OnMessageHandler(object sender, MessageEventArgs e) {
    Debug.Log("WebSocket server said: " + e.Data);
}
```

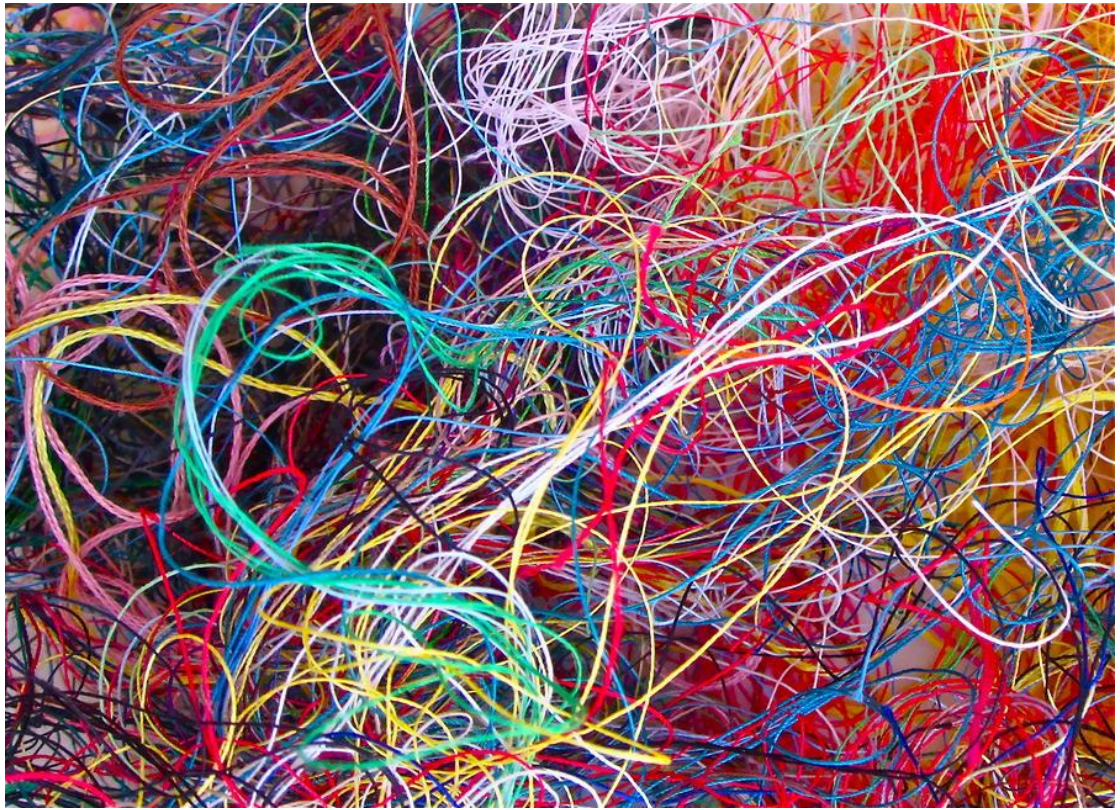
Disconnecting

```
private void OnCloseHandler(object sender, CloseEventArgs e) {
    Debug.Log("WebSocket closed with reason: " + e.Reason);
}

ws.CloseAsync();
```

Live Demo!

But wait, asynchronous means...



THREADS!

Unity threading model

- Single-threaded with coroutines
- API generally not thread-safe
- Anything that updates the screen must happen on the main thread

websocket-sharp threading model

- .NET threads
- Anything done asynchronously happens on a background thread
- Event handlers (OnOpen, OnMessage, etc.) are invoked on background threads

And never the twain shall meet?

- This would be a (more) boring talk
- Enter our old Computer Science buddy, the Finite State Machine

A simple state machine

```
using UnityEngine;

public enum State { NotRunning, Running, Connected, Ping, Pong, Done }

public delegate void Handler();

public class StateMachine : MonoBehaviour {
    public void Run() { … }
    public void Transition(State state) { … }
    public void AddHandler(State state, Handler handler) { … }
    public void Update() { … }
}
```

Requesting transitions

```
using System.Collections.Generic;

public class StateMachine : MonoBehaviour {
    private readonly object syncLock = new object();
    private readonly Queue<State> pending = new Queue<State>();

    public void Transition(State state) {
        lock(syncLock) {
            pending.Enqueue(state);
        }
    }
}
```

Transition handlers

```
public class StateMachine : MonoBehaviour {
    private readonly Dictionary<State, Handler> handlers
        = new Dictionary<State, Handler>();

    public void AddHandler(State state, Handler handler) {
        handlers.Add(state, handler);
    }
}
```

Handling transitions

```
public class StateMachine : MonoBehaviour {
    public void Update() {
        while (pending.Count > 0) {
            currentState = pending.Dequeue();

            Handler handler;
            if (handlers.TryGetValue(currentState, out handler)) {
                handler();
            }
        }
    }
}
```

Start me up!

```
public class StateMachine : MonoBehaviour {
    private State currentState = State.NotRunning;

    public void Run() {
        Transition(State.Running);
    }
}
```

Putting it all together (1)

```
using UnityEngine;
using WebSocketSharp;

public class StatefulMain : MonoBehaviour {
    public StateMachine stateMachine;
    private WebSocket ws;

    void Start() {
        ws = new WebSocket("ws://echo.websocket.org");

        ws.OnOpen += OnOpenHandler;
        ws.OnMessage += OnMessageHandler;
        ws.OnClose += OnCloseHandler;
```

Putting it all together (2)

```
        stateMachine.AddHandler(State.Running, () => {
            ws.ConnectAsync();
        });
        stateMachine.AddHandler(State.Connected, () => {
            stateMachine.Transition(State.Ping);
        });
        stateMachine.AddHandler(State.Ping, () => {
           ws.SendAsync("This WebSockets stuff is a breeze!",
OnSendComplete);
        });
        stateMachine.AddHandler(State.Pong, () => {
            ws.CloseAsync();
        });
```

Putting it all together (3)

```
private void OnOpenHandler(object sender, System.EventArgs e) {
    Debug.Log("WebSocket connected!");
    stateMachine.Transition(State.Connected);
}
private void OnMessageHandler(object sender, MessageEventArgs e) {
    Debug.Log("WebSocket server said: " + e.Data);
    stateMachine.Transition(State.Pong);
}
private void OnCloseHandler(object sender, CloseEventArgs e) {
    Debug.Log("WebSocket closed with reason: " + e.Reason);
    stateMachine.Transition(State.Done);
}
```

Putting it all together (4)

```
void Start() {
    // WebSocket init code from previous slides

    // State machine handlers from previous slides

    stateMachine.Run();
}
```

Live Demo!

Use the Source, Luke

https://github.com/jmglov/unity-ws-demo