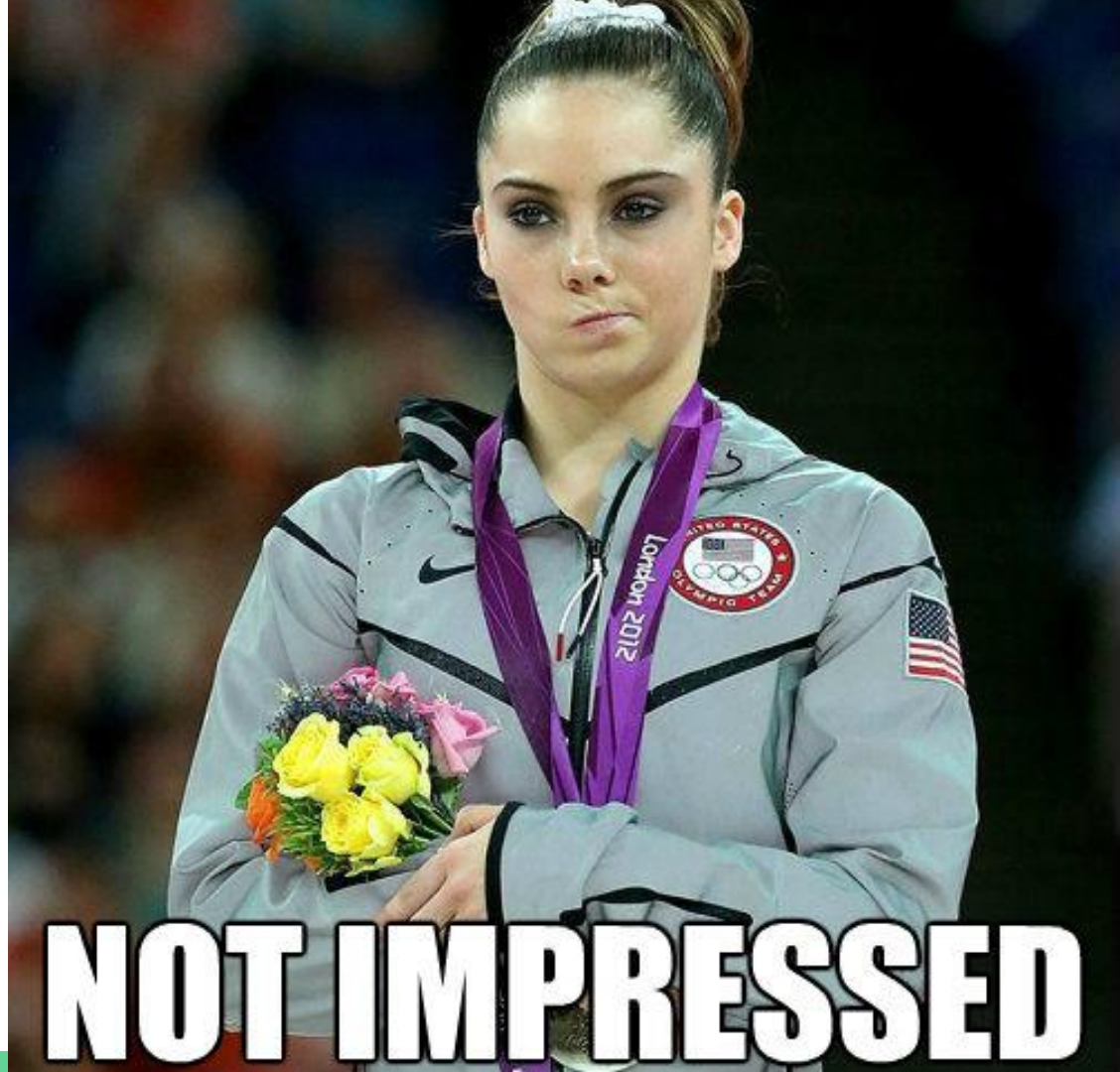
WHATWG-stream ALL the THINGS

And get off the main thread?

While working on RTCQuicTransport

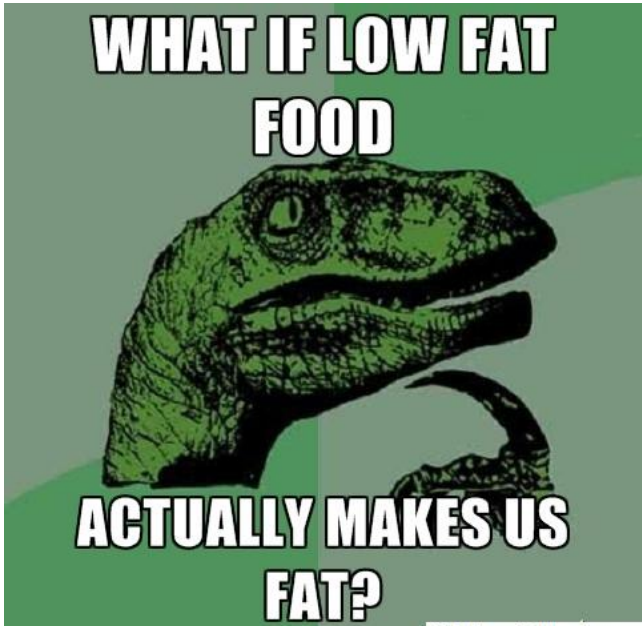
People kept asking for WHATWG streams

At first I was ...



But maybe it's not so bad for QUIC data channels

```
interface RTCQuicTransport {
  RTCQuicStream createSendStream();
  RTCQuicStream createBidirectionalStream();
  ReadableStream<RTCQuicStream> receiveStreams();
  WritableStream<ReadableStream> sendStreams();
}

interface RTCQuicStream {
  readonly attribute ReadableStream? readable;
  readonly attribute WritableStream? writable;
  ...
}
```



But it's sad being lonely

Will others come along too?

SCTP data channels?

```
partial interface RTCDataChannel {
  WritableStream sendStream();
  ReadableStream<ReadableStream> receiveStreams();
}
```

MSE?

```
partial interface SourceBuffer {
  WritableStream appendStream();
}
```

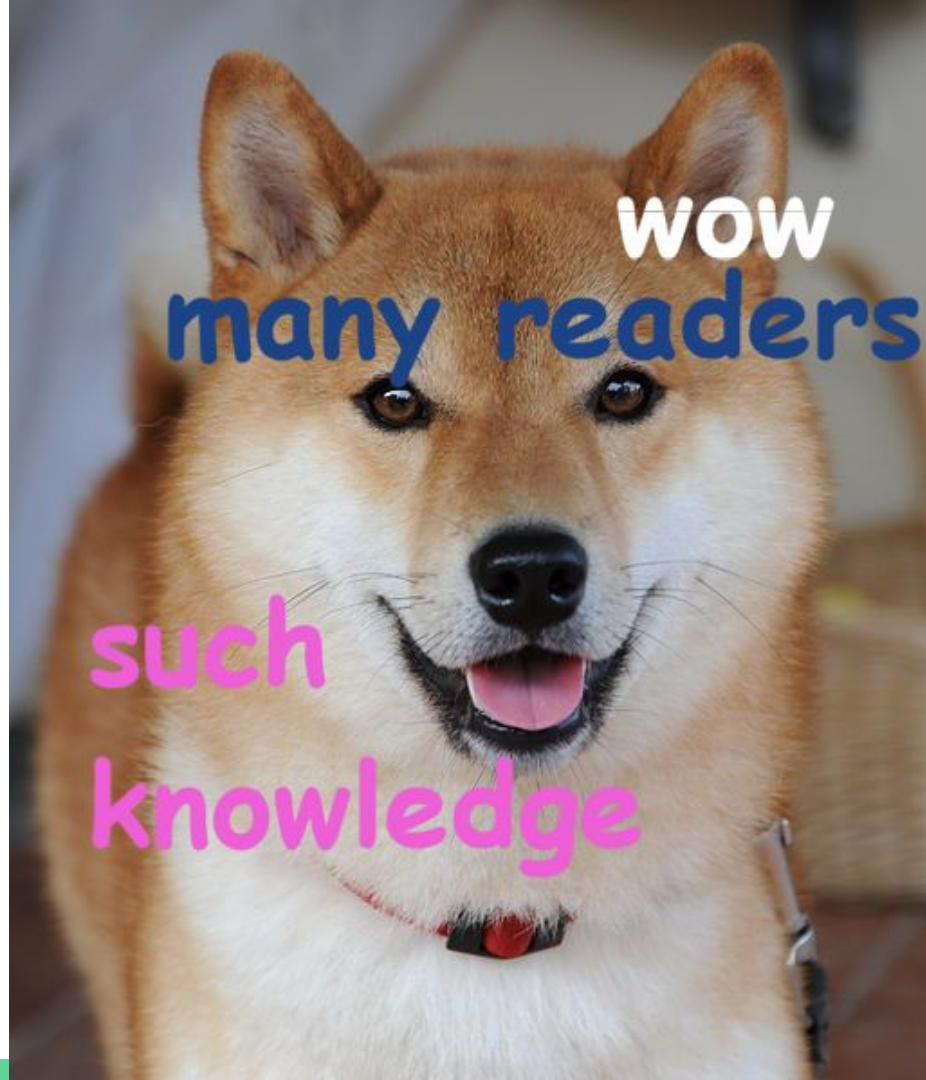
RTP?

```
partial interface RtpSender {
  ReadableStream<EncodedMediaOrFeedback> readEncodedFrames();
  WritableStream<EncodedMediaOrFeedback> writeReceiverFeedback();
}

partial interface RtpReceiver {
  WritableStream<EncodedMediaOrFeedback> writeEncodedFrames();
  ReadableStream<EncodedMediaOrFeedback> readFeedback();
}
```


Then we could really connect some pipes

SCTP to MSE?

```
var sctp = ...;  // SCTP RTCDataChannel
var mse = ...;  // MSE SourceBuffer
var parser = transformStream(parseMessage);

sctp.readStreams()
  .pipeThrough(parser)
  .pipeInto(mse.appendStream())
```

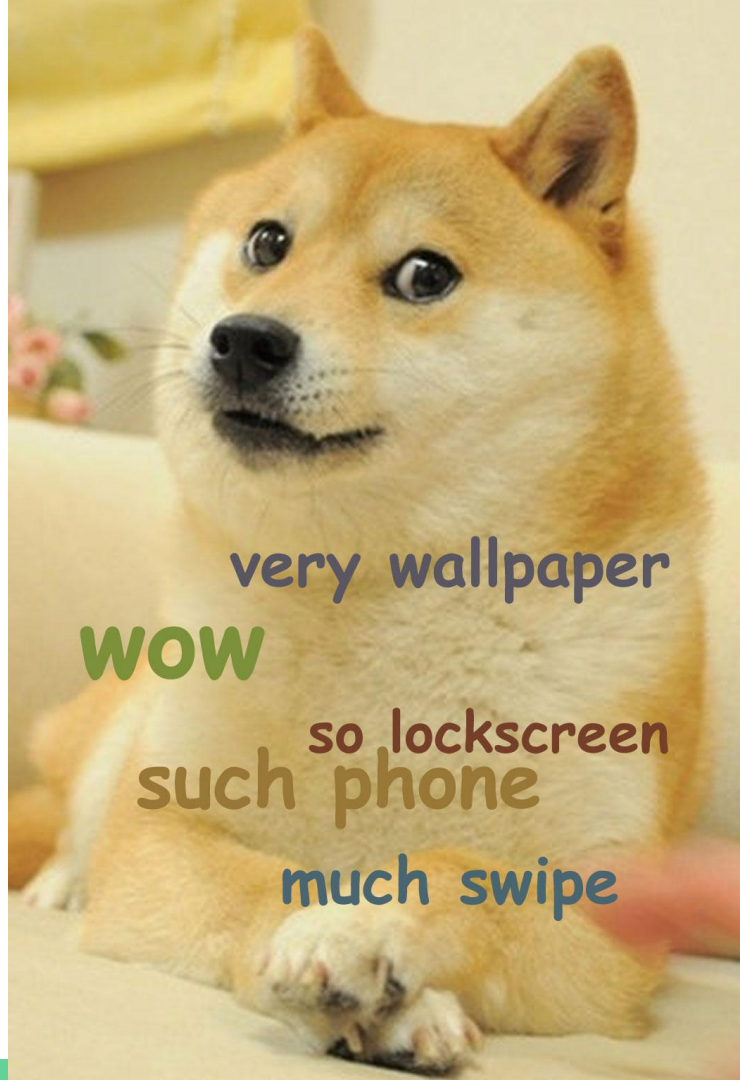


QUIC to MSE?

```
var quic = ...;  // RTCQuicTransport
var mse = ...;  // MSE SourceBuffer
var parser = transformStream(parseMessage);

quic.receiveStreams()
  .pipeThrough(parser)
  .pipeInto(mse.appendStream())
```

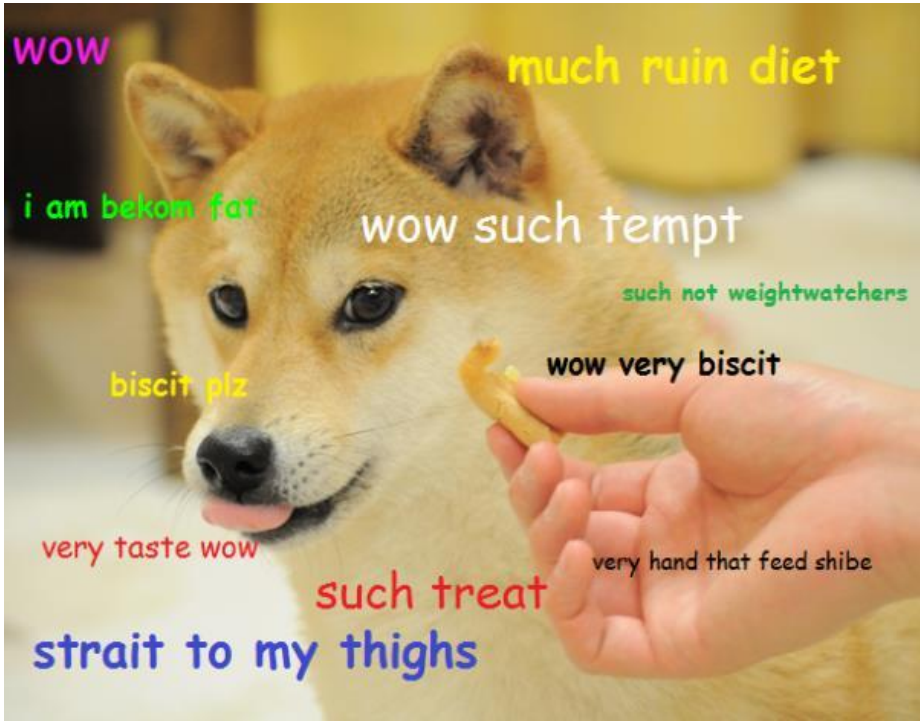
QUIC to WebRTC?

```
var quic = ...;  // RTCQuicTransport
var receiver = ...; // RtpReceiver
var parser = transformStream(parseMessage);
var serializer = transformStream(serializeMessage);

quic.receiveStreams()
  .pipeThrough(parser)
  .pipeInto(receiver.writeEncodedFrames())
receiver.readFeedback()
  .pipeThrough(serializer)
  .pipeInto(quic.sendStreams());
```



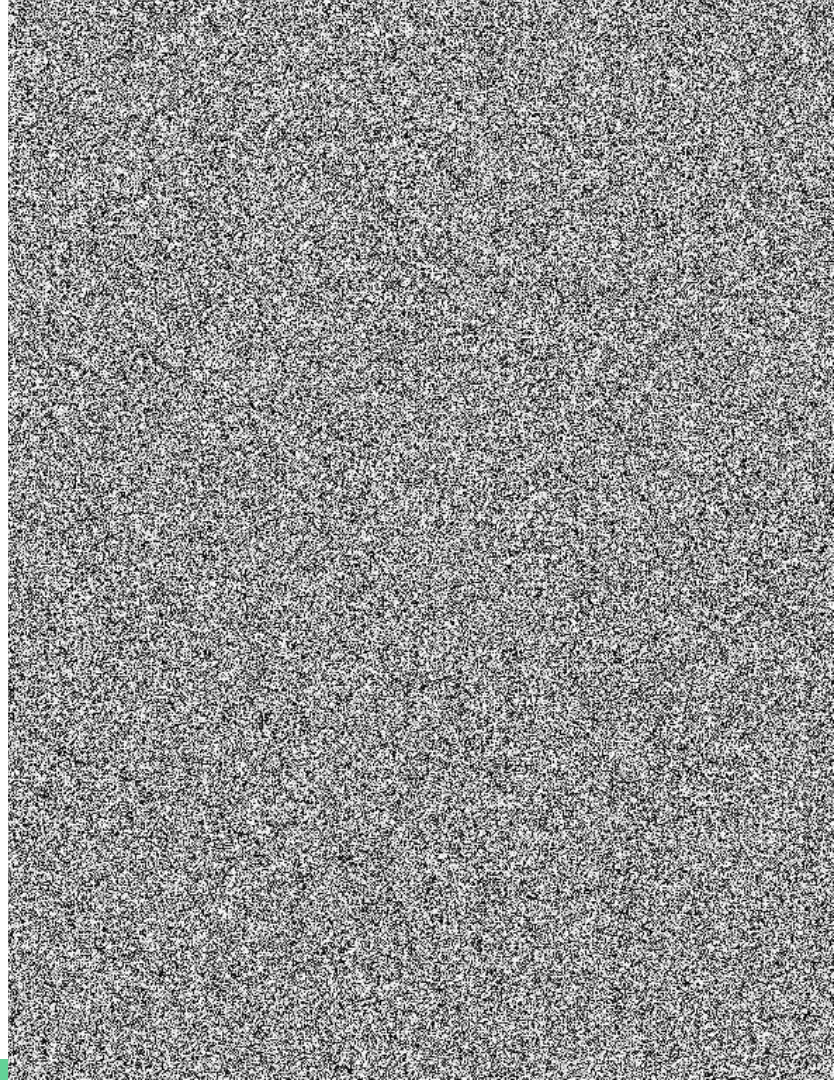
WebRTC to QUIC?

```
var quic = ...;  // RTCQuicTransport
var sender = ...;  // RtpSender
var serializer = transformStream(serialize);
var parser = transformStream(parseMessage);

sender.readEncodedFrames()
  .pipeThrough(serializer)
  .pipeInto(quic.sendStreams());
quic.receiveStreams()
  .pipeThrough(parser)
  .pipeInto(sender.writeReceiverFeedback())
```



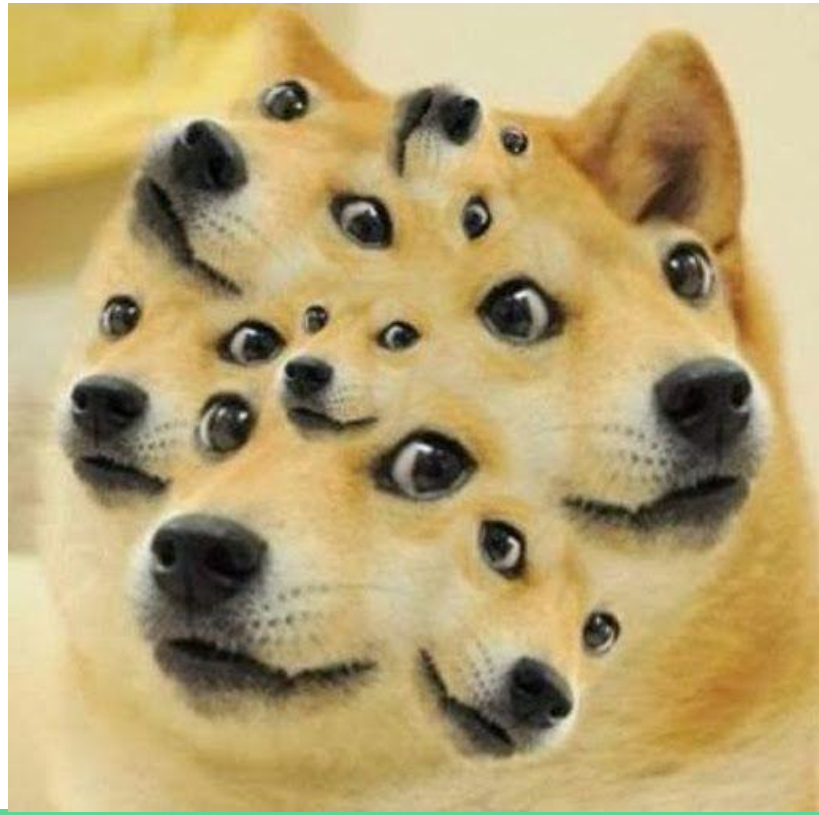
E2EE

```
var sender = ...;  // RtpSender
var receiver = ...;  // RtpReceiver
var encryptor = transformStream(encryptFrame);
var decryptor = transformStream(decryptFrame);

sender.readEncodedFrames()
  .pipeThrough(encryptor)
  .pipeInto(sender.writeEncodedFrames());

receiver.readEncodedFrames()
  .pipeThrough(decryptor)
  .pipeInto(receiver.writeEncodedFrames());
```



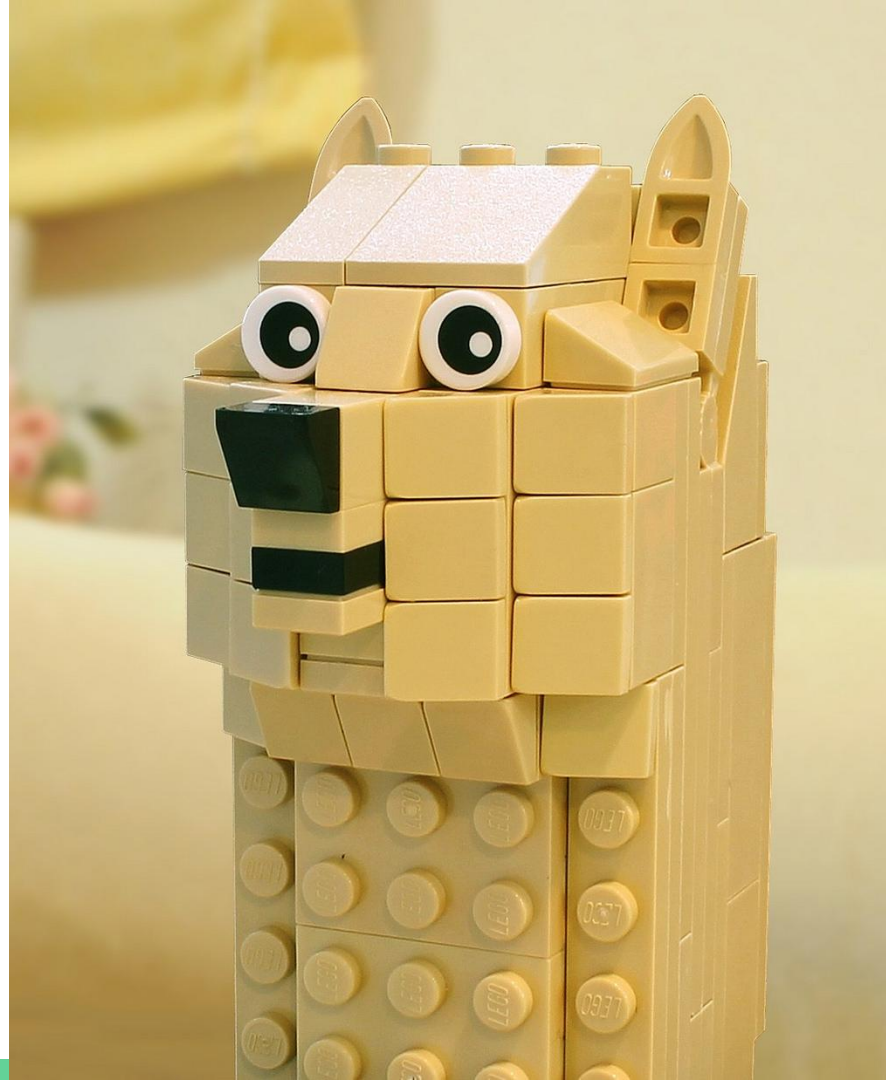
RTP to RTP? (without transcoding)

```
var sender = ...;  // RtpSender
var receiver = ...;  // RtpReceiver

receiver.readEncodedFrames()
  .pipeInto(sender.writeEncodedFrames());
```



RTP to MSE? (history!)

```
var rtp = ...;  // RtpReceiver
var mse = ...; MSE SourceBuffer
var serializer = transformStream(serialize);

rtp.readEncodedFrames()
  .pipeThrough(serializer)
  .pipeInto(mse.appendStream());
```



But you can already do this without streams

The real value comes from

Getting off the main JS thread

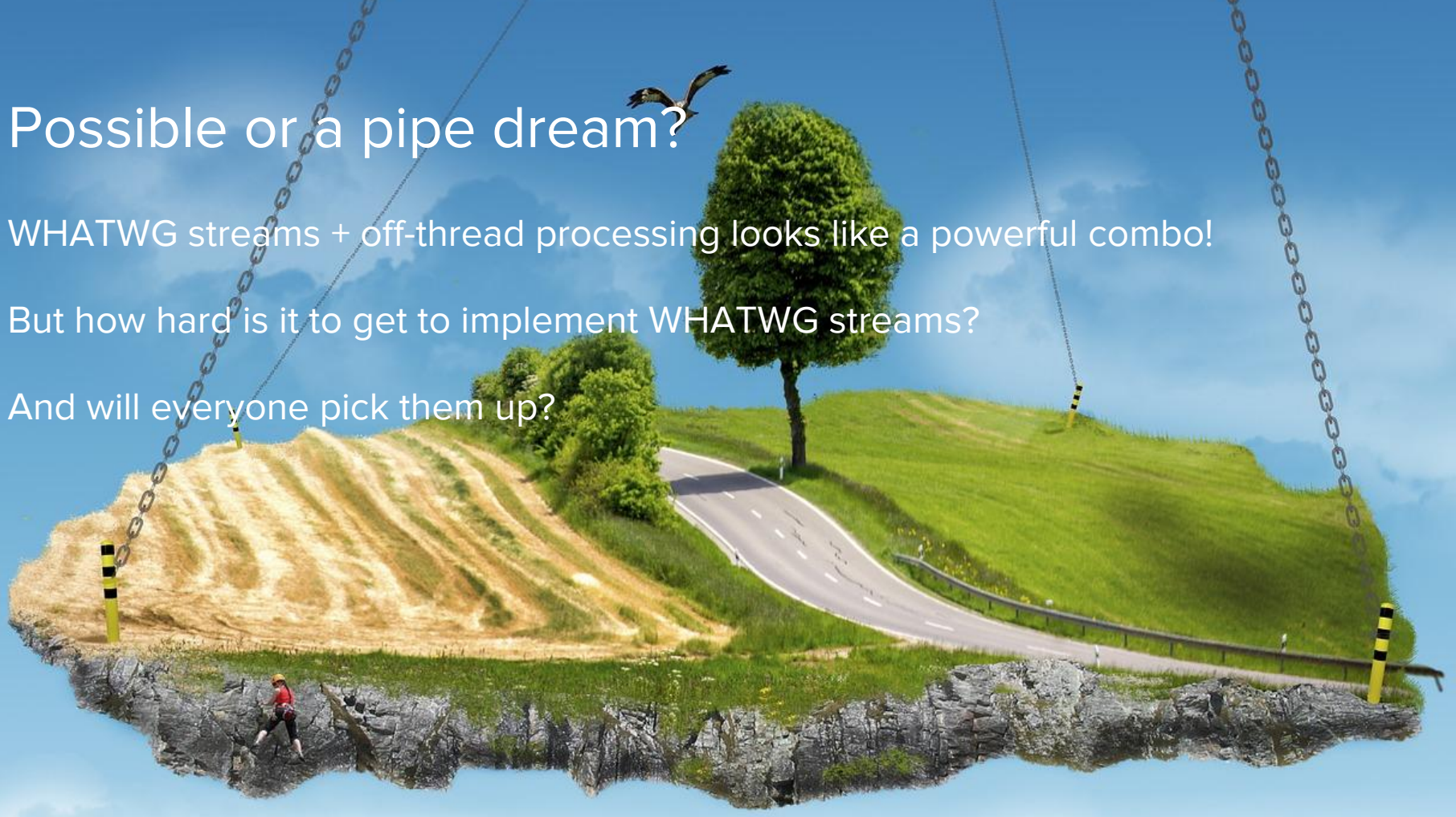
(click the link for more slides)

Possible or a pipe dream?

WHATWG streams + off-thread processing looks like a powerful combo!

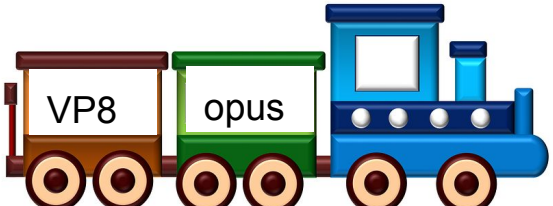
But how hard is it to get to implement WHATWG streams?

And will everyone pick them up?



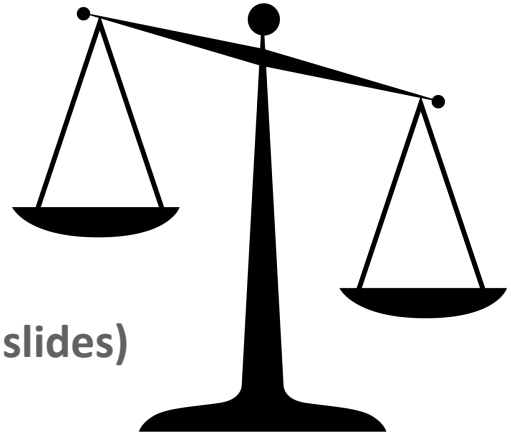
What would the stream chunks be?

```
typedef MediaOrFeedback EncodedMediaFrame or
RTCKeyFrameRequest or RTCMediaStats

interface EncodedMediaFrame {
  attribute DOMTimeStamp timestamp;
  attribute DOMString mimeType;
  attribute SourceBuffer encodedData;
}

interface EncodedAudioFrame : EncodedMediaFrame {
  attribute unsigned short channels;
  attribute unsigned long samplesPerSecond;
  attribute unsigned long samples;
}
```




```
interface RTCKeyFrameRequest {
  attribute DOMTimeStamp? lastDecodedTimestamp;
}

interface RTCMediaStats {
  attribute RTCTrackSenderStats? senderStats;
  attribute RTCTrackReceiverStats? receiverStats;
}

interface EncodedVideoFrame : EncodedMediaFrame {
  attribute DOMTimeStamp? duration;
  attribute bool keyFrame;
  attribute unsigned short rotation;  // in degrees
}
```

Pros/Cons of WHATWG streams vs not



`better:`

- **piping, esp. if we can get off the main thread (see upcoming slides)**
- piping, esp. if other things adopt streams
  - `receiveStream.pipeThrough(parser).pipeInto(...)`
  - `transport.receiveStreams.pipeThrough(parser).pipeInto(...)`

`worse:`

- **Uncertain maturity, adoption, performance, ease of impl of streams**
- Can't write fin bit and last chunk at the same time
- More clunky for simple things
  - `receiveStream.readable.getReader().read(view).then(...)`
  - `transport.receiveStreams.getReader.read().then(...)`

Conclusion

It seems that switching to using WHATWG streams would be better if:

- We can also get something like WorkletTransformStream or web workers to pipe things off the main thread
- We can also get things to pipe to/from (RTP, MSE)
- We can implement this whole thing in a performant way