WebRTC 1.0 objects

at 2015 f2f day 2

Update on PeerConnection.connectionState (PR 278)

New state added. The rules are now:

1. If all "new" or "closed" => "new"
2. else if any "failed" => "failed"
3. else if any "connecting" or "checking" => "connecting"
4. else if any "disconnected" => "disconnected"
5. else (all "connected" or "completed" or "closed") => connected

Update on RtpEncodingParameters (PR 273)

- Removed resolutionScale
- Removed framerateScale (until simulcast is figured out)
- Made framerateBias an enum of
  - maintain-framerate
  - maintain-resolution
  - maintain-balance
- Renamed framerateBias to degredationPreference

Better name ideas are welcome :)

WebIDL of degredationPreference

```
dictionary RTCRtpEncodingParameters {
  RTCDegredationPreference degredationPreference;
  // ...
}

enum RTCDegredationPrefernce {
  maintain-framerate,
  maintain-resolution,
  maintain-balance
}
```

Example of degredationPreference

```
encoding.degredationPreference = "maintain-framerate";
```

Codec reordering (PR 298)

```
dictionary RTCRtpParameters {
  // ...
  sequence<RTCRtpCodecParameters> codecs;  // These can be reordered or removed.
}

dictionary RTCRtpCodecParameters {
  // These are all read-only.
  DOMString mimeType;
  unsigned long clockRate;
  unsigned short channels;
  DOMString sdpFmtpLine;
}
```

Codec selection updated (PR 298)

- To unset: sender.setParameters({encodings: [{..., payloadType: undefined}]});
- If renegotiation blows away the payloadType: automatically unset (RtpSender keeps sending, but with a different codec.

Codec reordering example

```
var params = sender.getParameters();
for (var i = 0; i < params.codecs.length; i++) {
  // Put thor at the front so that it's preferred.
  if (params.codecs[i].mimeType == "video/thor") {
    params.codecs.unshift(params.codecs.splice(i, 1));
    break;
  }
}
```

What about RtpSender.getCapabilities()?

```
partial interface RTCRtpSender {  // Same as RTCRtpReceiver
  static RTCRtpCapabilities getCapabilities(DOMString kind)
}

dictionary RTCRtpCapabilities {
  sequence<RTCRtpCodecCapability> codecs;
  sequence<RTCRtpHeaderExtensionCapability> headerExtensions;
}

dictionary RTCRtpCodecCapability {
  DOMString mimeType;
}

dictionary RTCRtpCodecCapability {
  DOMString uri;
}
```

I thought of a use case

```
if (RtpSender.getCapabilities().codecs.some(c => c.mimeType ==
"video/h264")) {
  showJoinConferenceButton();
} else {
  showMessage("Sorry, your browser is incompatible");
}
```