Proposal

We keep to our general approach of errors, exceptions and error events.
We try to align as closely as possible to the DOM specification of these objects.

GetUserMedia spec:

- Change the NavigatorUserMediaError to be a subclass of DOMError. This means just that the “name” attribute is specified in the superclass. “message” is referred to, but not defined in, DOMError, so its status seems unclear. If this is deemed to be present, the only extra attribute is “constraintName”, which is gUM specific.
- Change the enum NavigatorUserMediaErrorName to have error names in the style of section 3.4 of the DOM spec: CamelCase with uppercase initial.
- Ask the DOM team to add PermissionDenied and ConstraintNotSatisfied to their error list (per their expressed desire). No corresponding code.

- In text, state that all exceptions thrown are DOMExceptions.
- Change all references to codes, like “INVALID_STATE_ERR”, to be the equivalent strings, like “InvalidStateError” (this is the only exception thrown in this spec.)

PeerConnection spec:

- Remove RTCError and make RTCSdpError a subclass of DOMError. Replace all references to RTCError with DOMError.
- Change the enum RTCExceptionName and RTCErrorName to have error names in the style of section 3.4 of the DOM spec, or remove them once SessionDescriptionError is registered in the DOM spec.
- Remove INVALID_CONSTRAINTS_TYPE, INVALID_CANDIDATE_TYPE, INVALID_MEDIASTREAM_TRACK. We should use the Javascript builtin TypeError instead, as per DOM guidance.
- Use DOM’s InvalidStateError instead of INVALID_STATE
- Use DOM’s InvalidModificationError instead of INCOMPATIBLE_CONSTRAINTS
- Ask the DOM team to register a SessionDescriptionError instead of INVALID/INCOMPATIBLE_SESSION_DESCRIPTION

- In text, state that all exceptions thrown are DOMExceptions, except for TypeError.

Background

Present spec: error handling

There are 3 different things in the spec that happen because of what we’d call errors. Some of them may happen for other reasons as well (like events signalling a non-error situation),

but let's focus on the error situation.

- Error callbacks are called when a specific request to the API results in an unsuccessful outcome. The normal case should then be that the state of the underlying object does not change, and the error happens exactly once per request.
- Exceptions are thrown when something bad happens in an API call. A typical case is the "type error" mismatch, usually generated from libraries (DOM Error 14). When an exception is thrown, the state of the object should be unchanged.
- Events are dispatched at the objects presented at the API in various situations. These events have to carry information enough to guide the app in what to do about them, and some of them need diagnostics - such as the SDP line or construct that caused an error in a SetLocalDescription call.

Being inventive is rarely good. If we can be consistent with other specs, programmers will have a better time learning our APIs, and will use them more consistently.

Other specifications we should align with

The most important specs to align with are the HTML5 spec and the DOM spec.
From the HTML5 spec, we inherit the whole Event mechanism including the EventHandler interface.

From the DOM spec, we should align with errors.
(discussion: http://lists.w3.org/Archives/Public/public-webrtc/2012Dec/0115.html)

DOMError is here:
https://dvcs.w3.org/hg/domcore/raw-file/tip/Overview.html#interface-domerror
It is intended for "specs that want to introduce error handling by other means than exceptions" - the only defined member is a name field.

the DOM spec also describes the EventTarget:
https://dvcs.w3.org/hg/domcore/raw-file/tip/Overview.html#eventtarget

DOM exceptions are here:
https://dvcs.w3.org/hg/domcore/raw-file/tip/Overview.html#exception-domexception
The core exception definition is here:
http://dev.w3.org/2006/webapi/WebIDL/#idl-exceptions

The DOM Exception has a "name" field and a "message" field, inherited from the WebIDL spec (http://dev.w3.org/2006/webapi/WebIDL/#idl-exceptions), in addition the IDL says that it has a "code" - it doesn't say what the code is when there is no code for a message.