# MediaCap and WebRTC Error Handling

Presenter: Harald Alvestrand

## Principles of the API

- Call a function
- Exactly one of three things happens:
    - An exception is thrown
    - A success callback is called
    - An error callback is called

# Current language

Errors are indicated in two ways: exceptions and objects passed to error callbacks. Both forms of error reporting MUST provide an object of type RTCError. An exception MUSTbe thrown in the following cases:

- The type of any argument passed to a function did not match what was expected. An appropriate string from the RTCExceptionName enum MUST be used as the error name.
- A function call was made when the RTCPeerConnection is in an invalid state, or a state in which that particular function is not allowed to be executed. In this case, the string INVALID_STATE MUST be used as the error name.

In all other cases, an error object MUST be provided to the failure callback. The error name in the object provided MUST be picked from either the RTCExceptionName orRTCErrorName enums.

## Desirable properties not found yet

- Behavior on illegal parameters should be defined to match what current APIs and compilers produce (DOM errors)
- It should be easy to tell what situations result in an exception and what in an error callback

## Alternative API structure

- Object returned from call
- Success and error events fired at structure
- Parameters stored in structure

This would be a structure similar to what IndexedDB is doing, while our current callbacks is more like what Geo is doing.

## Emulating "alternate" in Javascript

```
function AltGetUserMedia(constraints) {
   obj = {};
   GetUserMedia(constraints,
       function(stream) {
           obj.stream = stream;
           if (obj.onsuccess) {
               obj.onsuccess();
           }
        },
        ....
   );
   return obj;
}
```

## Evaluating this change proposal

- Advantage is not clear
- Developers have to be familiar with both patterns
- It's easy to mask one pattern with the other, if desired
- Change is disruptive (*)

Proposed decision: No change.

(*) insert argument about the wisdom of developers who expect unfinished standards not to change, and temper with the realization that we depend on them to tell us if our approach works

## New API points - design

- We are adding new API points to the API
  - GetStats
  - Recorder
- Need a principled decision for these
  - Consistent with existing: Callbacks
  - Consistent with other WGs' design: Status objects
  - No consistency needed
- Thoughts?