Finding an Ideal Algorithm

We'll ideally finish this today

Principles

1. **Keep it simple**. If it's complex, we might never finish.
2. **It doesn't have to be perfect.** If you want more, use "advanced".

"min distance" algorithm

1. Enumerate the modes across multiple cameras (or other devices)
2. Filter out modes which are `< min`, `> max`, or `!= exact` constraints
3. Apply any "advanced" constraints to further filter out modes.
4. Select the mode with the minimum distance from the `ideal` values, calculated as the sum different types:
    - `CLOSE (height, width, aspectRatio): |actual-ideal|/ideal`
    - `CLOSE_OR_GREATER (framerate): (actual > ideal) ? 0 : ((ideal-actual)/ideal)`
    - `MATCH (facingMode) actual == ideal ? 0 : 1`
    - `STRONG_MATCH (sourceId) (actual == ideal) ? 0 : 1000000`
5. When distances are equal, the implementation is free to choose.
6. Other or future constraints can choose one of the types.

"min distance" example

```
Constraint: { width: { ideal: 1280 },
              height: { ideal: 720, min: 360, max: 1080 },
              aspectRatio: { ideal: 1.78 }}

Camera Modes:
320x240,  // Filtered out
640x360,  // Distance = 1 = |1280-640|/1280 + |720-360|/720 + |1.78-1.78|/1.78
640x480,  // Distance = 1.08 = |1280-640|/1280 + |720-480|/720 + |1.78-1.33|/1.78
800x600,  // Distance = 0.8 = |1280-800|/1280 + |720-600|/720 + |1.78-1.33|/1.78
854x480,  // Distance = 0.666 = |1280-854|/1280 + |720-480|/720 + |1.78-1.78|/1.78
960x540,  // Distance = 0.5 = |1280-960|/1280 + |720-540|/720 + |1.78-1.76|/1.78
1280x720,  // Distance = 0 = |1280-1280|/1280 + |720-720|/720 + |1.78-1.78|/1.78
1920x1080  // Distance = 1.0 = |1280-1920|/1280 + |720-1080|/720 + |1.78-1.78|/1.78
```

"min distance" qualities

- simple to spec
- simple to understand
- good results
- testable
- consistent across browsers
- easy enough to implement in a spreadsheet:

https://docs.google.com/a/google.com/spreadsheets/d/1HFP39jHcd3cI-a3maEj1_QzNyKzRBiZ0isge55yBpIs/edit#gid=0

"min distance" full text

1. Enumerate all of the possible modes of cameras and microphones (or other capture devices).
2. Filter out all of the modes which have a property which is less than a "min" constraint, greater than a "max" constraint, or not exactly equal to an "exact" constraint.
3. Apply any "advanced" constraints to further filter out modes which do not fit the advanced constraints.
4. For the remaining modes, select the mode which minimizes the "distance" from the ideal, where "distance" is calculated by the sum of the "ideal" constraint values (`|a|` meaning the absolute value of a), choosing from one of the following distance formulas:
    - CLOSE (height, width, aspectRatio, sampleSize): `|actual-ideal|/ideal`
    - CLOSE_OR_GREATER (framerate, sampleRate): `(actual > ideal) ? 0 : ((ideal-actual)/ideal)`
    - MATCH (facingMode, echoCancellation): `actual == ideal ? 0 : 1`
    - STRONG_MATCH (sourceId): `(actual == ideal) ? 0 : 1000000`
5. Future constraint properties can choose a formula.

"min distance" things to consider

- When does "advanced" get applied? Before or after? It seems like after is impossible.
- When distances are equal (such as when there are no "ideal" values at all), the browser is free to choose. Is that OK?
- We threw in some extra weight for sourceId. Was that a good idea? Should we also throw in some weight for framerate or

"advanced expansion" algorithm

Turn this:

```
width: { ideal: 40 },
height: { ideal: 60 },
```

Into this:

```
{advanced: [
  { height:{min:60,max:60}, width:{min:40,max:40}},
  { height:{min:50,max:70}, width:{min:30,max:50}},
  { height:{min:40,max:80}, width:{min:20,max:60}},
  { height:{min:60,max:60}, width:{min:10,max:70}},
  … and so on ...
]}
```

"advanced expansion" rules

1. "exact", "min", and "max" constraints must be satisfied, and are not expanded.
2. "ideal" numbers are expanded to ranges starting from the "ideal" value and expanding by a given "step" out to the min and max.
3. "ideal" strings and enums are expanded to a bare value.
4. Merge the advanced lists (ranges and bare values) into one big advanced list by taking one constraint from each advanced list and putting it into a set which is added to the merged advanced list.
5. Choose a set of step values, one for each non-string constraint type. And also choose a default min and max for the expanding ranges for when min or max are not given.

"advanced expansion" visual

aspect ratio	height	width	framerate
1.3333	720	1280	30
1.233 - 1.433	710 - 730	1270 - 1290	29 - 31
1.133 - 1.533	700 - 740	1270 - 1290	29 - 31
1.033 - 1.633	690 - 750	1270 - 1290	29 - 31
0.933 - 1.733	680 - 760	1270 - 1290	29 - 31
...	...	...	...
0.5 - 2.0	240 - 1080	320 - 1920	15 - 60

"advanced expansion" steps

	step	default min	default max
height	10	240	4320
width	10	320	7680
aspectRatio	.1	0.25	4.0
framerate	1	10	60
volume	0.05	0.0	1.0
sampleRate	8,000	8000	192,000
sampleSize	2	8	32

Which algorithm?

- "min distance"?
- "advanced expansion"?
- none (implementation-specific)?

We (Peter, Justin, Cullen, Jan-Ivar, Dan, Stefan) recommend

"min distance"

Are we done?

Ideally, we are done and have choose the algorithm with the minimum distance from the ideal algorithm.