Measure/Smart Units

Ben Allen

24 October 2024

~~Measure: Inspired by Smart Units~~
Measure: its own thing

- The proposal under consideration here was originally inspired by Smart Units
- However! There has been significant interest in Measure in contexts far afield from internationalization
- This raises questions about what to include

Design problem: preview

- We absolutely *must* be able to represent mixed units
- (feet and inches, meters and centimeters)
- Mixed units present API design problems!
- How do we take mixed-unit input?

Properties

- Providing an object to keep the following pieces of data together:

1. Measured values
2. The units in which they're measured
3. The precision at which they're measured

Methods

- `new Measure(value, unit, precision)`

- `convert(unit, precision)`
- `localeConvert(locale, usage)`
- `split()`

(`precision` **and** `usage` **optional**)

Precision and mixed units

- Take precision as number of fractional digits, or as number of significant figures? Fractional digits seems like the correct choice for Smart Units, but not for Measure
- Default precision should be high?
- Precision needs to adhere to the *smallest* unit in the Measure.

Precision and mixed units

- Take precision as number of fractional digits, or as number of significant figures? Fractional digits seems like the correct choice for Smart Units, but not for Measure
- Default precision should be high?
- Precision needs to adhere to the *smallest* unit in the Measure.

Design problems: mixed units ruin everything beautiful

- `new Measure(value, unit, precision)`

Okay, so how do we specify mixed units?

proposal: We could use the names given by CLDR in units.xml:

`"foot-and-inch", "meter-and-centimeter", "pound-and-ounce"`

Design problems: mixed units ruin everything beautiful

alternate proposal:

- Take an array of units `["foot", "inch"]` **for** `unit`
- **And likewise for an array** `[6, 0]` **for** `value`

Advantage: Allows users to specify conversions not given in units.xml
Disadvantage: How do we deal with incoherent combinations?

```
new Measure([6, 0], ["feet", "cups-metric"])
```


Design problems: mixed units ruin everything beautiful

alternate alternate proposal:

- Take an array of value-unit pairs
- This makes it visible that precision adheres to the entire Measure, i.e. only the smallest unit.

```
new Measure([{value, unit}], precision)
```


Design problems: "per" units ruin everything beautiful

- *x* per *y*: how do we handle this?
- If we use CLDR unit names, we don't have to worry about it – units.xml specifies all relevant x-per-y values / conversion factors
- If we support arbitrary *x* per *y* units, what does the user input?
- (I dislike supporting arbitrary *x* per *y* units, but that is largely irrelevant)

Design problems: mixed units ruin everything beautiful

- `new Measure(value, unit, precision)`

Okay, so how do we specify mixed units?

proposal: We could use the names given by CLDR in units.xml:

`"foot-and-inch", "meter-and-centimeter", "pound-and-ounce"`

What units to include?

Smart Units: probably should not include units of energy?
Measure: probably should include units of energy?

- Reasons we limited Smart Units:

1. Received feedback about how discoverability could be a problem if large number of units included
2. Previous versions of the proposal proposed a *very* large and somewhat nebulously defined set of units. This received very negative feedback

What units to include?

- Okay, so: if we're proposing to create this Measure object that knows how to do a wide range unit conversions, do we really have a justification for making Smart Units more limited?

- (probably so, but I wanted to raise the issue anyway?)

More mixed unit questions

- `convert(unit, precision)`

1. Should we take an array of units?
2. (… or we could just use CLDR's unit names…)

Methods

`localeConvert(locale, usage)`

- Whew! This one's pretty straightforward. We take a locale, we return a Measure localized to that locale
- Take an array of locales, in case we need fallbacks?
- `usage` is optional – needed by Smart Units

Methods

`split()`

- Also straightforward! Returns an array of value/unit pairs representing each unit (one item for non-mixed units)

Thank you!