Locale Extensions

Ben Allen

9 September, 2023

igalia

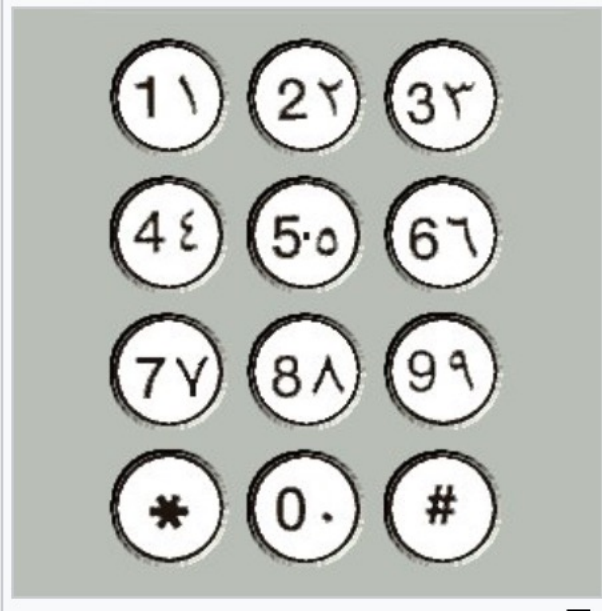
Three interrelated problems

1. Oftentimes there are multiple numbering systems used in one locale with no easy way for users to indicate which they prefer

The problem #1

Some regions have multiple commonly-used number systems

- **`hi` defaults to** ”latn”, even though many people requesting that locale might prefer ”deva”
- Competing number systems are used in **the United Arab Emirates, among other Middle Eastern/South Asian countries.**



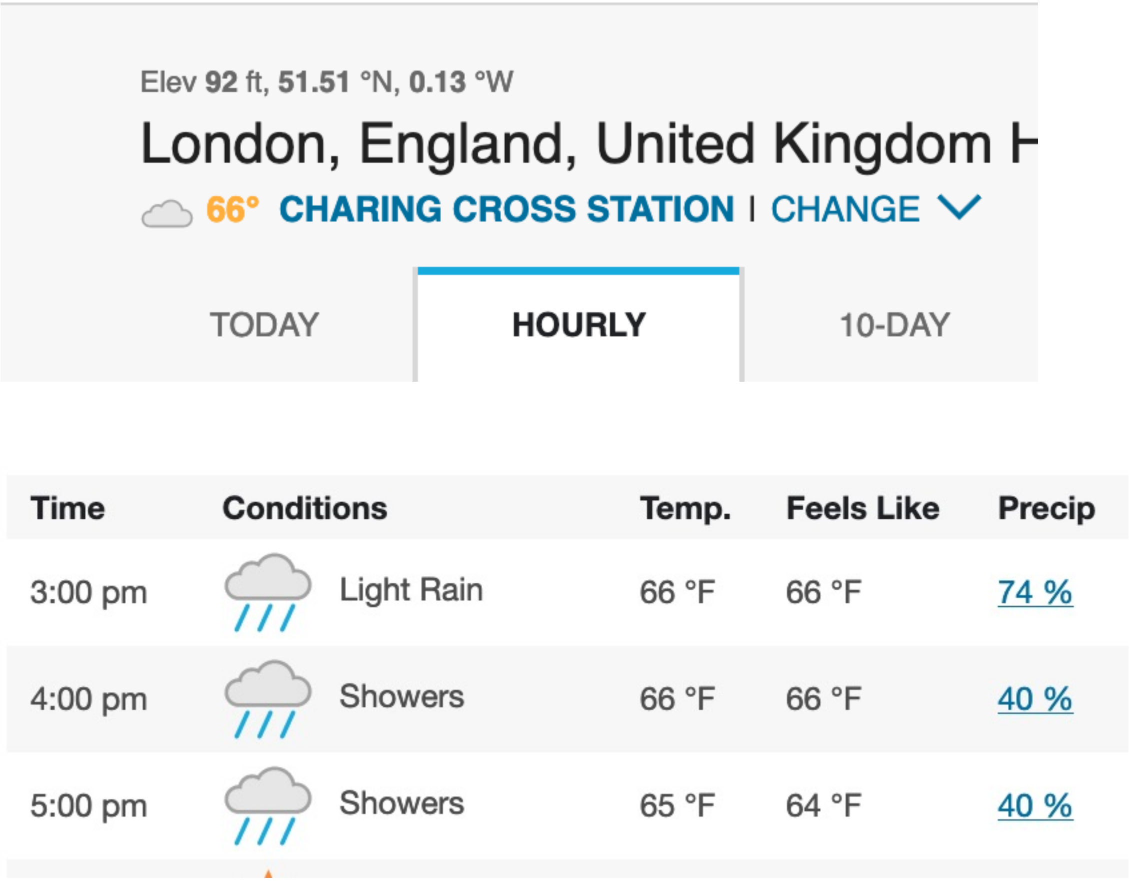
(telephone keypad with both Eastern Arabic and Latin numerals)

The problem #2

Often users will have content tailoring desires that differ from the defaults used for the locale the content they're viewing is in.

A plurality of sites only offer content in English, and often in a regional variant of English with highly idiosyncratic defaults for hours of the day, temperature measurement, first day of week, etc.

Users might want to view these things in a more globally common way.



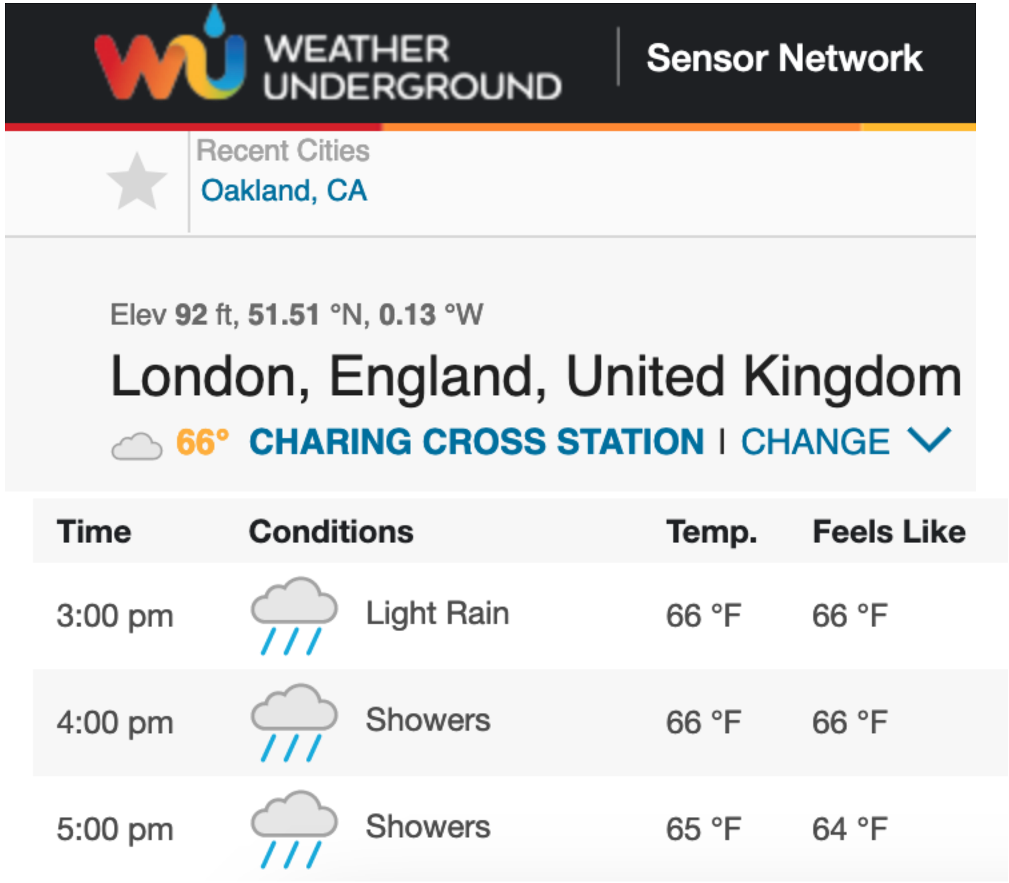
Elev 92 ft, 51.51 °N, 0.13 °W

London, England, United Kingdom H

66° CHARING CROSS STATION | CHANGE

Time	Conditions	Temp.	Feels Like	Precip
3:00 pm	Light Rain	66 °F	66 °F	74 %
4:00 pm	Showers	66 °F	66 °F	40 %
5:00 pm	Showers	65 °F	64 °F	40 %

The problem #2



WEATHER UNDERGROUND | Sensor Network

Recent Cities

Oakland, CA

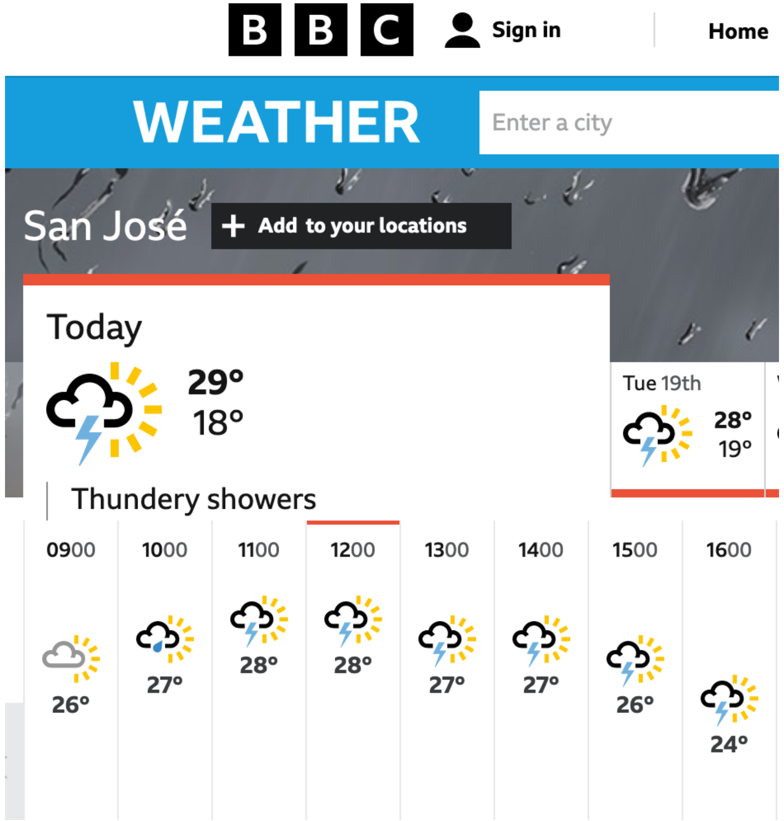
Elev **92** ft, **51.51** °N, **0.13** °W

London, England, United Kingdom

66° **CHARING CROSS STATION** | CHANGE

Time	Conditions	Temp.	Feels Like
3:00 pm	Light Rain	66 °F	66 °F
4:00 pm	Showers	66 °F	66 °F
5:00 pm	Showers	65 °F	64 °F

The problem #2
(the other way around)



San José

Today

29°
18°

Thundery showers

Tue 19th

28°
19°

0900	1000	1100	1200	1300	1400	1500	1600
26°	27°	28°	28°	27°	27°	26°	24°

The problem #3

Often users will have content tailoring desires that differ from the defaults used for the locale the content they’re viewing is in.

3. Some users have combinations of preferences that differ from both the default for the locale specified in their OS and also the default for the content they’re viewing
    - Someone who wants content tailored for `en-US`, except they want calendars to show Monday as the first day of the week.

Currently

Currently these preferences are ignored

- Browsers can read the OS locale to determine the system language, and convey that to servers via `Accept-Language`
- If content is not available in any of the requested languages, server falls back to another language. Content tailoring is as in the defaults for that language; any region-specific tailorings the user might have wanted *even if* the server couldn't get them their preferred language are ignored.
- Not good! The client has given the server potentially identifying information, and has gotten no content tailoring at all in return.

The problem within the problem

Fingerprinting opportunities abound!

- Users who list multiple languages in `Accept-Language` are likely making themselves immediately individually identifiable.
- Were we to provide a mechanism for clients to directly convey **all** their OS preferences to servers, this would also likely make any users with non-default settings **immediately** individually identifiable.

Our goal:

1. Let users express their content tailoring preferences as fully as possible
2. While prioritizing tailorings that might seriously impact content legibility/intelligibility if ignored
   - (most notably: numbering system)
3. While leaving as small a fingerprinting surface as possible
4. Ideally, *smaller* than the surface offered by one item in an `Accept-Language`.
5. Afingerprinting that happens must be detectable – no passive fingerprinting!

Not our goal

1. Allowing users to express arbitrary preferences
   - This is a fingerprinting nightmare!
2. Making web applications that are as flexible as native applications
   - Not possible – the internet is a hostile place
3. Finding a way to avoid revealing any information at all
   - No revealed information -> no localization

How close can we get?

How close can we get to complete localization without making users individually identifiable?

1. We could allow support for something like the `–u-rg` tag:
   - Allow users to express the concept that regardless of what locale the content they receive is, they would like that content tailored to match the first language in the `Accept-Language` header.
   - This doesn’t reveal anything more than what’s revealed in `Accept-Language`, but current privacy best practices say that “this is already revealed elsewhere” is not a valid defense for adding features that provide fingerprinting vectors

How close can we get?

How close can we get to complete localization without making users individually identifiable?

1. We could *separate out individual components* of the preferences that could be expressed by `-u-rg`, and send the individual components.
   - This is a **giant** gain for some very common use cases: people preferring content tailored as in `en-US`, people preferring content tailored for global standards.
   - Enough people want those collections of preferences (representable as "-u-fw-mon-hc-h23-mu-celsius" and "-u-fw-sun-hc-h12-mu-fahrenhe") that if people could ask for them, they'd be able to hide in the crowd.

How close can we get?

How close can we get to complete localization without making users individually identifiable?

1. We could *separate out individual components* of the preferences that could be expressed by `-u-rg`, and send the individual components.
    - Possibly even a gain for people with preferences that fall between the two (for example, people using the defaults for es-MX, which can be represented as “-u-fw-sun-hc-h12-mu-celsius”), provided enough others use that set of preferences.
    - **User research is required.**

“User research is required”

- “User research is required” is a phrase that shows up a lot in this talk.
- Depending on results of user research, aspects of the annoyingly complex solution to come may not be necessary.

Annoying complication #1

The scheme outlined in the explainer is annoyingly complicated, since there’s three factors directly involved with content localization in this context:

1. The number of people who share those preferences
2. Who are (through Accept-Language or implicitly through geolocation, etc.) requesting content
3. **and are given content in a particular locale.**

The combination of preferences reflecting the defaults in ‘es-MX’ might be widely used in `en-US` and not used at all in (for example) `fi`

- Someone accessing content in `fi` and requesting tailorings as in the default for `es-MX` is likely making themselves immediately individually identifiable.

Annoying complication #2

People may have *very important* preferred content tailorings that don't match the defaults for *any* locale. Key example: users wanting a secondary numbering system for their locale, for example "hi-u-nu-deva"

- We **must** account for this case – it directly impacts content intelligibility

A complication that's not our problem: people might have highly idiosyncratic preferences, for example, preferring temperatures in Kelvin.

- It's reasonable for us to ignore this preference, because there's no way we can honor it without making the user fingerprintable.
- No internationalization-related reason to allow this preference

Proposed solution

- For each locale, determine via **user research** what sets of **default preferences for other locales** might be safely expressed without unnecessary reductions in the size of each user's anonymity set
- For each locale, determine via **user research** what, if any, alternate preferences for that locale might be in common use (i.e. the 'hi-u-nu-deva' example)
- Why default preferences for locales?
  1. it's less difficult to measure than measuring bespoke preferences
  2. Likely the only preferences commonly used are either defaults in another locale or alternates for the current locale.

Proposed solution

- Why “for each locale”?
    - A combination of preferences that lets you hide in one locale might make you immediately identifiable in another.

Proposed solution: numbering system

- `-u-nu` is the highest priority extension to allow
- Consider *always* allowing users to select the numbering system designated as “native” for their locale?

Proposed solution

During implementation:

- Implementers determine what combination of preferences are likely safe in each locale.
- These are included with each revision of the browser.

During use:

- Browser reads OS preferences from system
- Browser determines (pick your favorite algorithm) which of those preferences are expressible through the available preference strings
- Only those preferences are revealed to the server

Settable preferences in current revision

In the current revision, these are the –u extension tags we’re concerned with:

- `ca`: calendar
- `fw`: first day of week
- `hc`: hour cycle
- `mu`: temperature measurement unit
- `nu`: numbering system

Preferences that correlate

- `fw`: first day of week
- `hc`: hour cycle
- `mu`: temperature measurement unit

These would be commonly used, tend to be relatively strongly correlated with each other, and allowing for them solves a lot of major annoyances for many users.

Rare preference that's important

- \`nu\`: numbering system

Much less commonly used than the previous three, and doesn't correlate particularly strongly with any other tag. However! Allowing users to express this preference is very important, because *not* allowing for it can result in unintelligible content

Rare preference that is also important

- `ca`: calendar

Somewhat awkward! Most of the world uses “u-ca-gregory”, and requesting anything else results in immediately revealing as much information as an `Accept-Language` header with only one language. (Do you want “-u-ca-buddhist”? I’m guessing you’re from Thailand.)

- Non-Gregorian calendars may be safely expressible in some locales – perhaps even ‘en-US’! – but not in many others
- Likely anyone who wants Gregorian can get it, regardless of what language the content is in

Preference you want to use consistently

- `mu`: temperature measurement

This one can produce mayhem if it’s available in some locales but not others. Converting from Fahrenheit to Celsius in your head is annoying, but not knowing what temperature scale a temperature is in is infuriating.

Mechanisms:

1. `Client Hint` headers for each of the five tags.
   - If a server has to explicitly request each one, it becomes more clear when they're gathering irrelevant data for fingerprinting.
   - (makes the fingerprinting vector an active fingerprinting vector)
2. A JavaScript API that can be used to discover settings for each of these five tags
   - or rather, for each of the five tags **that are actually expressible using a safe -u extension string**
   - Others left undefined
   - Settings must be requested **individually** – as with `Client Hint`s, this prevents passive fingerprinting.

Potential problem

- We want to allow all users to express that they want one of the alternate numbering systems for their region.
  - (really, it should be part of the non-extended language tag, like script is)
- However, allowing the expression of this preference can be dangerous if those settings strongly correlate to the user being a member of a politically oppressed group
- Decisionmaking requires taking into account not just raw numbers of users, but also geopolitical conditions.

User research may show opportunities for simplification

Thank you!