https://w3c.github.io/IFT/Overview.html, reviewed for TPAC although most of the suggestions don't materially change anything and don't need to be dealt with at TPAC.

References to Dominik feedback refer to Review Incremental Font Transfer Spec.

Created from a quick pre-read to prep for TPAC f2f, take with salt.

Abstract

Nit: could it have an anchor so we can link to it easily?

Perhaps we could spare a word to note what's special about incxfer? That is, that it doesn't trade efficient transfer against damage to rendering fidelity. Exploratory rewrite:

This specification defines a method to incrementally transfer a font from server to client. The client loads only the portion(s) of the font that they actually need, significantly reducing data transfer. Multiple incremental additions to the same font are possible, such as a user agent updating a font as a user browses multiple pages. Incremental transfer improves on unicode-range by avoiding damage to layout (kerning, ligatures, etc) rules, meaning it can efficiently support fine grained increments to latin and complex scripts like Indic or Arabic.

https://w3c.github.io/IFT/Overview.html#intro
"Without this technology, a browser needs to download every last byte of a font before it can render any characters using that font" - seems confusing in the face of unicode-range. Technically correct because each @font-face of unicode-range is a standalone font but that's a pretty subtle distinction. Too subtle for an intro IMHO.

"The success of WebFonts is unevenly distributed" - +1 to Dominik's comment that this seems to need elaboration. Perhaps extends to the whole paragraph, the woff2 for CJK bit seems to ignore that woff2 and unicode-range play particularly nicely for CJK (or any other script sthat doesn't use layout)

https://w3c.github.io/IFT/Overview.html#overview
Nit "...is a regular OpenType font..." - no change in meaning if you delete the word "regular", avoids the potential implication of being the Regular style.

"Partly in virtue of" reads a bit curiously. Spitballing,

> (current) "An *incremental font* is a regular OpenType font that is reformatted to include incremental functionality, partly in virtue of two additional tables."

→

(updated) "An *incremental font* is an OpenType font with some or all glyph data removed and metadata added to describe how to fetch incremental additions *(patches [link to definition]*) to restore the removed content when the client needs it."

The definition of *incremental font* needs to be more clearly established. We say it's an OT font reformatted with new tables and and then in "At a high level..." we refer to an "initial font file" that seems to be what we just referred to as an *incremental font* in the beginning of the overview. And then a Font Subset ambushes us in an alley.

A small, well defined, set of terms seems like it would help a lot. More on that below.

https://w3c.github.io/IFT/Overview.html#making-incremental-fonts
+1 to Dominik's comment about creating curiosity about uncommon ways

Here "incremental font" seems to refer to the complete set of initial font file and patches? The first paragraph of Overview reads to me as suggesting it specifically means the initial font file.

Seems a bit detailed for an Overview. I wonder if it should move to some other section, with the overview giving only a high level description of the idea that an encoder can produce a set of files served as "normal" http assets that are readily deployable using any web server or CDN. That ease of use aspect really isn't coming through for me right now.

Suggest describing the process of simple operation in a little more detail, e.g. a web site operator generates a bunch of files from a single OpenType font then references the "root" from CSS, the browser intelligently loads fragments as needed without the kind of damage to rendering fidelity unicode-range introduces.

https://w3c.github.io/IFT/Overview.html#performance-considerations
Good stuff but ... is it really overview material?

Feels like maybe there should be a non-normative section on how users would operationalize incxfer that discusses the notion of a font ⟶ incremental font + patches fileset, perf considerations, etc?

https://w3c.github.io/IFT/Overview.html#opt-in
I feel like Example 1, or an additional Example 2, should actually show multiple URIs. That is, show some of the things currently mentioned in NOTE blocks. Might also wish to comment on interaction with unicode-range, e.g. that multiple @font-face in a unicode-range could each individually incremental transfer.

Hm. Also what about a case where the user wants *one* incremental transfer @font-face or *many* unicode-range ones, that is, to fallback to unicode-range. Can that be spelled in CSS? - merely having tech on src urls doesn't seem sufficient. **Update** per f2f at TPAC this is achieved by putting the incremental @font-face as the highest priority unicode-range. We should add this example.

The section title is a bit obtuse. It certainly is an opt-in mechanism but ... perhaps we could mention that it's about how to declare you want to use incxfer in css? "CSS @font-face updates" as the heading and opt-in, offline use as 2.1, 2.2?

https://w3c.github.io/IFT/Overview.html#definitions
I feel like *incremental font* should be defined here?

Terminology feels a bit like it's jumping around. Overview told me we'd have a *incremental font* to which we apply *patches*. And yet... we don't define *incremental font* and *Font Subset* snuck up on me from somewhere. Feels more difficult to read for someone new to the domain than it has to be, perhaps we could narrow down to a small set of relatively intuitive terms?

For example, https://w3c.github.io/IFT/Overview.html#font-patch-definitions refers to an "IFT-encoded font" which appears to be a new and undefined term. I think it's a potentially good term. For example, what if we standardized on:

- IFT font: an OpenType font with additional metadata describing the available patches that can be fetched to augment its capability.
    - An "initial font" is just an IFT font with the least capability it will ever have. Could be as little as 0 glyphs although it doesn't have to be, that's up to the operator.
    - If Overview talked about requesting an augmenting an IFT font ... that would make sense.
    - Producing an IFT font from a "normal" (not-IFT) OT font would also make sense.
- Patch: the unit of augmentation.

Consider axing Font Subset and Font Patch.

Codepoints, layout features, feature tags, and design space all merit definition even if it's just a link to some other place that defines (such as layout and code points).

For example, 4.3. Incremental Font Extension Algorithm refers to (code points, feature tags, design space). Edit: 4.2. Default Layout Features references text shaper, that probably makes sense to define as well?

Consider moving definitions above "Opt-In Mechanism"

https://w3c.github.io/IFT/Overview.html#font-subset-dfn
This refers to (code points, layout features, variation axis space)
The algorithm: section refers to (code points, feature tags, design space)

https://w3c.github.io/IFT/Overview.html#extending-font-subset
We refer to *patches* as the unit of change to an *incremental font*. Then we get here and start talking about extending font subsets. The terminology feels a bit muddled. Perhaps it would be more consistent to use the same terms, e.g. "4. Extending a Font Subset" → "4. Applying a patch." Or if you take the IFT font definition about, "Augmenting an IFT font with a patch" or some such?

https://w3c.github.io/IFT/Overview.html#font-patch-invalidations
I think it's trying to tell me that I can't necessarily apply more than one patch to the same IFT font. Could say so more clearly IMHO. Also doesn't really get into implications, e.g. if you need 2 patches on the same IFT font you might have to fetch one, then look at the updated patch map and see which, if any, to fetch now.

https://w3c.github.io/IFT/Overview.html#extend-font-subset
Would love to see this adopt fewer terms. E.g. instead of font subset we have an "IFT font" (or incremental font ... exact name doesn't matter as much as having fewer, more consistent, terms)

The inputs to this algorithm are:

- *IFT font: the current state of the IFT font, with 0..N patches already applied*
- *IFT font URI: ...*
- *Capability request: the*

Font subset definitions doesn't feel intuitive, holdover from the very hb-subset oriented original idea? Suggest a name that more clearly indicates this is added functionality the client wants. "Patch request" or "Client demand" or "Capability request" or something like that?

The algorithm:

"Remove any entries in entry list which have a patch URI which was loaded and applied previously during the execution of this algorithm." – how is the client supposed to know this? Is it computable from the IFT font or does the client have to persist additional state such as a list of patch URIs that have been loaded? - that might not fit caching well. If it's merely URIs fetched during the current execution of the algorithm I suggest explicitly including *visited: the*

URIs of patches fetched during this execution of the algorithm, initially an empty set as an input and describing how it is updated throughout the process.

Rationale to limit #patches seems missing?

Steps 6 and 7 could be collapsed by all three current bullets say "if entry list contains one or more" and adding a fourth bullet for "Otherwise return the unmodified IFT font"

Handle errors

Suggest this should encourage the client to use the font, consistent with how unicode-range works (if some font-face urls fail to load we'd still use the ones that worked)

https://w3c.github.io/IFT/Overview.html#fully-expanding-a-font
Here too I wish it was defined in terms of an *incremental font* or an *IFT font* (preferred) rather than a *font subset*. Font subset is over-general and not specific to IFT.

Speculation: I wonder if we should just have a hook to provide a url to the whole font? - to go offline fetch whole-font-uri and call it a day.

https://w3c.github.io/IFT/Overview.html#caching-incremental-fonts
I think a little more exposition wouldn't hurt. Explicitly touch on how conceptually you augment the IFT font and update *the same cache entry*. Any future access should use the updated entry.

https://w3c.github.io/IFT/Overview.html#patch-map-format-1
compatibilityId specifies a *random* value but that doesn't actually appear necessary, it need only be a value that has not been used previously. Monotonic increasing would be entirely valid. Remove the word "random" unless I'm missing something. If there is a reason (e.g. security) I missed it.

Feature Map encoding: "with entryMapRecords[0] corresponding to ..." is rather painful to read. Would it be simpler expressed in terms of how to find the jth entry of featureRecords[i]? E.g. Look at [sum of featureRecords[0..i-1).entryCount + j] ?

https://w3c.github.io/IFT/Overview.html#interpreting-patch-map-format-1
"Convert the set of glyph indices to a set of Unicode code points using the code point to glyph mapping in the cmap" - suggest being more explicit that we conceptually reverse the cmap to produce a gid:set-of-codepoints map and then union the set-of-codepoints we lookup for each of our gids to produce a final set of codepoints.

"Compute mapped entry index..." - this is also a bit painful to read. I wonder if we could describe it in terms of how to form a loop over the values and get a simpler result?

5.2.1.2. Remove Entries from Format 1
I would add a few words about why, e.g. "This algorithm is used to remove entries from a format 1 patch map *to ensure patches that have been applied are excluded when evaluating subsequent requests for capability"*

https://w3c.github.io/IFT/Overview.html#patch-map-format-2
Same feedback as ^ wrt random compatibilityId

defaultPatchEncoding - description seems like it could be simpler? - The default patch format, used unless a patch entry explicitly specifies otherwise