

Technical Disclosure Commons

Defensive Publications Series

June 2021

Human Readable Representation of Data

Tommy Nyquist

Siddhartha Sivakumar

Salvador Guerrero Ramos

Oystein Eftevaag

Follow this and additional works at: https://www.tdcommons.org/dpubs_series

Recommended Citation

Nyquist, Tommy; Sivakumar, Siddhartha; Ramos, Salvador Guerrero; and Eftevaag, Oystein, "Human Readable Representation of Data", Technical Disclosure Commons, (June 03, 2021)
https://www.tdcommons.org/dpubs_series/4349



This work is licensed under a [Creative Commons Attribution 4.0 License](https://creativecommons.org/licenses/by/4.0/).

This Article is brought to you for free and open access by Technical Disclosure Commons. It has been accepted for inclusion in Defensive Publications Series by an authorized administrator of Technical Disclosure Commons.

Human Readable Representation of Data

ABSTRACT

Software often generates lengthy and difficult-to-recall alphanumeric identifiers, e.g., b23fc5a0a2bd3c964910c47f6a3252bffa146. Examples of such identifiers include links to documents or web-resources, entries in databases, digital signatures, etc. The length and the seemingly arbitrary construction of the identifier makes it difficult for a human user, e.g., a programmer, to identify, memorize, or compare. This disclosure describes techniques that map lengthy alphanumeric strings to easy-to-recall icons and/or colors. The map can be a deterministic, many-to-one function, e.g., a hashing function.

KEYWORDS

- Emoji representation
- Hieroglyph
- Pictograph
- Emoticon
- Screen reader
- Assistive technology

BACKGROUND

Software often generates lengthy and difficult-to-recall alphanumeric identifiers, e.g., b23fc5a0a2bd3c964910c47f6a3252bffa146. Examples of such identifiers are widely found, e.g., links to documents or web resources, entries in databases, digital signatures, etc. The length and the seemingly arbitrary construction of the identifier (e.g., the use of non-language words or phrases) makes it difficult for a human user, e.g., a programmer, to identify, memorize, or compare identifiers.

For example, consider the following identifiers:

133da63f3fb7c7113a4e81c3872557db

ee0069752106eb64ccf607c136f0fd5f

4f1b2d80005930928b134560750927db

ab745b8dad1cda502390940dd5e7758c

Fig. 1 illustrates some techniques to display the above items on a screen in a somewhat human-readable manner.



Fig. 1: Shorter representations of identifiers

The representation of Fig. 1(a) does shorten the identifiers, but does so to a point that risks conflation. Depending on the screen they are displayed on, their order might be inconsistent. It might not be mathematically possible to consistently assign long identifiers to small numbers (such as 1, 2, 3, 4): two materially different identifiers may be assigned the same number. A different screen, with a different data query, may be unable to uniquely use a short identifying number. The scheme of Fig. 1(a) is infeasible.

The representation of Fig. 1(b) leverages the uniqueness of the identifiers to shorten them to a substring, e.g., to their last four characters. Although a 1:1 map between long and short identifiers is no longer guaranteed, the relatively large space of four-character strings is an assurance that two materially different long identifiers clash only rarely. However, although the

scheme of Fig. 1(b) is feasible, even the shorter identities are hard to memorize or compare across screens.

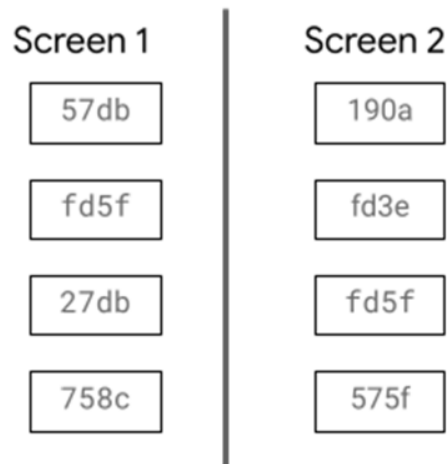


Fig. 2: Comparing short identifiers across screens is still strenuous for a human user

Fig. 2 illustrates the difficulty of comparing short identifiers across screens. It is not immediately apparent to the human eye that the second identifier of screen 1 matches the third one of screen 2; indeed, there is, at first glance, a false match may be perceived between the first identifier of screen 1 and the fourth one of screen 2.

Other techniques to make data identifiers more human readable are as follows. Some photo-sharing websites enable users to describe their uploaded photo in descriptive language, e.g., <http://example.com/image/NiceGreenWagon> instead of <http://example.com/image/c02c056d8c42460743e7418d152eb9e5a5025b88>. Similar techniques have been used to identify the same item across multiple screens. For example, across screens, identifiers that end in 3 are rendered in green, while identifiers that end in 4 are rendered in red.

DESCRIPTION

The length of the identifier is partly due to the potentially very large space of data that the identifier indexes. However, it is often the case that there is only a small subspace that the identifier effectively accesses. This disclosure describes techniques that leverage the observation that identifiers typically index only a small subspace of the space of all possible identifiers.

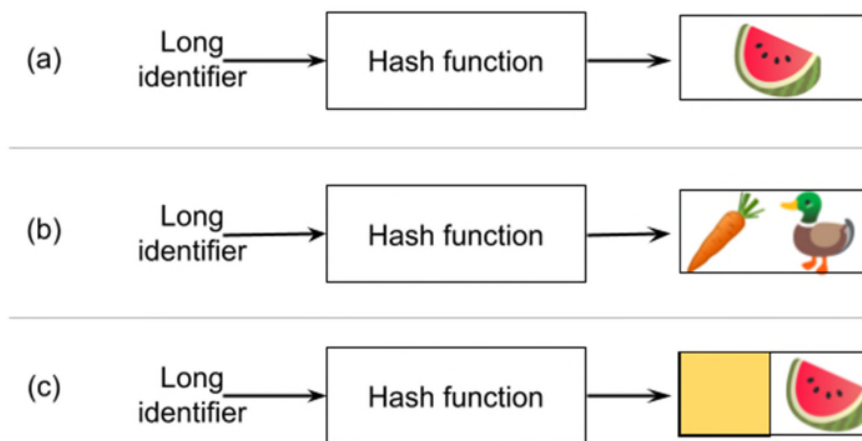


Fig. 3: Using a hash functions to map a lengthy identifier to easy-to-remember emojis and/or colors

As illustrated in Fig. 3, the described techniques map lengthy source data, e.g., lengthy identification strings, to easy-to-remember icons (e.g., emojis) and/or colors. The map can be a deterministic, many-to-one function, e.g., a hashing (checksum or cryptographic hash) function. In Fig. 3(a), a lengthy identifier is hashed deterministically to a smaller code, e.g., one of approximately eighteen hundred emojis available at present. Effectively, all lengthy identifiers are classed into one of eighteen-hundred picture-varieties. Such picture-varieties effectively work as mnemonics, and are easier for the human mind to recognize, memorize, and/or compare.

If eighteen-hundred emojis are an insufficient alphabet size, then one can use the technique of Fig. 3(b), where a lengthy identifier is hashed deterministically into a pair of emojis. The total emoji-alphabet size increases to 1800^2 , more than three million.

Alternatively, as illustrated in Fig. 3(c), a lengthy identifier can be hashed deterministically into a combination of a color and an emoji. Although in theory there are about 16.7 million colors (assuming a red, green, blue admixture, each taking one of 256 values), many are indistinguishable to the human eye. Restricting colors to web-safe ones, or named X11 colors, the number of distinguishable colors are in the high tens, such that an alphabet comprising color-emoji pairs has a size of about 180,000 — ample for most purposes.

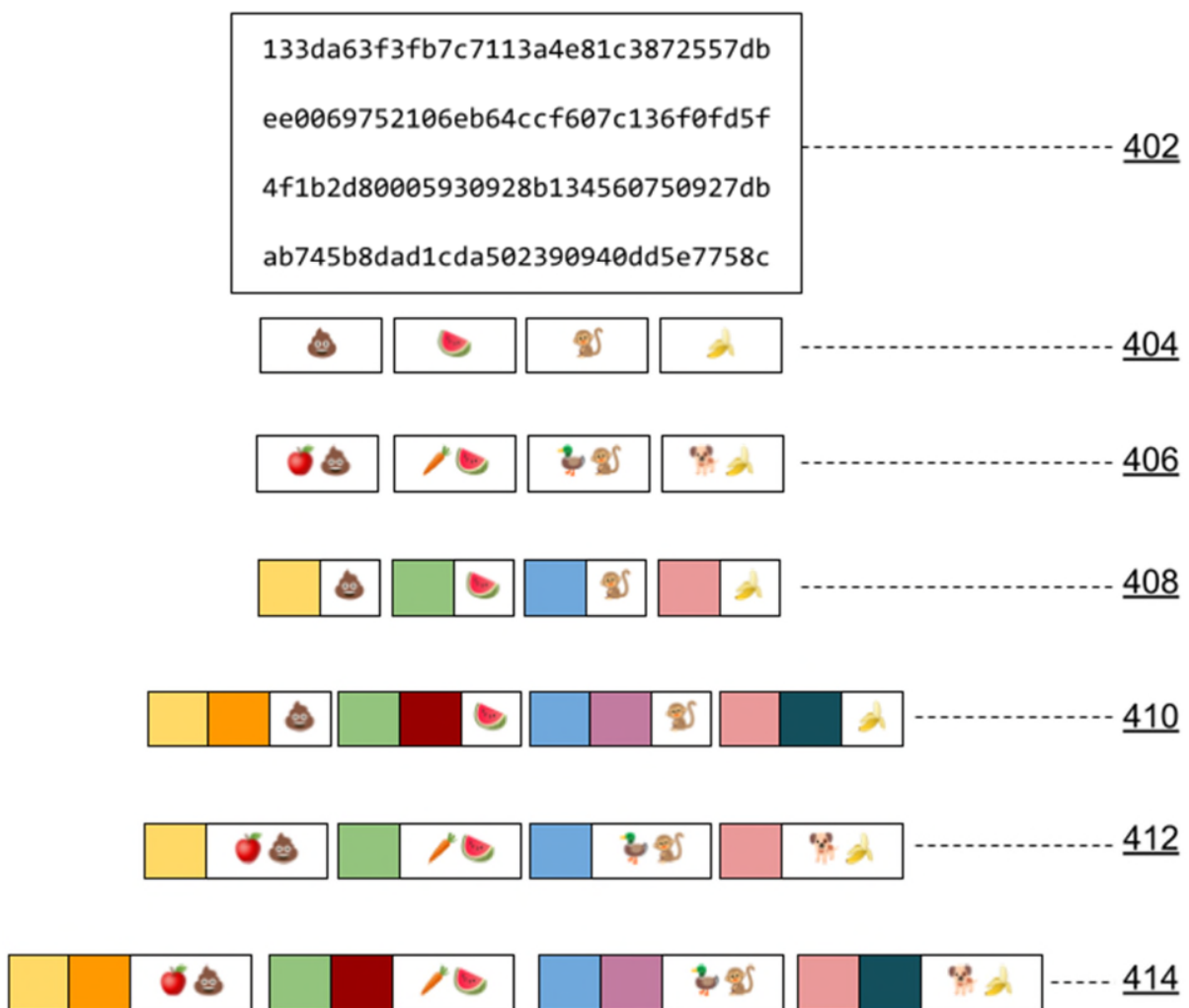


Fig. 4: Mapping identifiers to an alphabet of emojis and/or colors

Fig. 4 illustrates that four lengthy, difficult-to-recall, alphanumeric identifiers (402) can be represented as single-emoji strings (404); double-emoji strings (406); color-emoji strings (408); color-color-emoji strings (410); color-emoji-emoji strings (412); color-color-emoji-emoji strings (414); etc. In contrast to Fig. 1, all of the representations of Fig. 4 are easier for a human to recall, memorize, and/or compare. In addition, the representations of Fig. 4 span an amply large subspace of all possible alphanumeric strings of a given length.

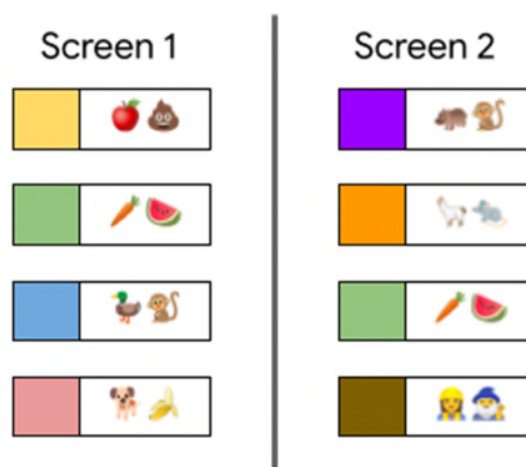



Fig. 5: A comparison of color-and-emoji based identifiers across screens

Fig. 5 illustrates a comparison of color-and-emoji based identifiers that a user might attempt across screens. Here, two application screens are visible at one time, and the user memorizes an identifier on one screen to find it on the other screen. In contrast to Fig. 2, the human eye can easily see that the second identifier of screen 1 is identical to the third identifier of screen 2, and that no other identifiers of either screen match.

Accessibility: Per the Unicode standard, every emoji has a language translation. For example, the emoji 🍉 has an English-language translation “watermelon,” while the emoji 🍌 is translated to English as “banana.” For individuals with impaired vision, e.g., those who rely on a screen reader to understand the information on the screen, emojis can simply be read off in their

language translations. For example, the identifier `ee0069752106eb64ccf607c136f0fd5f` of Fig. 4, which is rendered in double-emoji script as 🥕🍉 (406), is read by a screen reader as “carrot, watermelon.” The same identifier, which is rendered in color-emoji script as 🍉 (408) in Fig. 4, is read by a screen reader as “green, watermelon,” or an equivalent alternative text, content description, or accessibility-name. Arguably, the described techniques of rendering lengthy identifiers as emojis are beneficial also to vision-impaired users, who no longer have to recall lengthy, alphanumeric identifiers; rather, only their English-language equivalents. Similarly, colorblind users can use the shape of the emojis, rather than their colors, to differentiate identifiers. They can also adjust the selection of colors based on the nature of their color blindness. For example, red-green colorblind users can set colors to red, yellow, and blue.

In this manner, the techniques of this disclosure enable humans to intuitively read, recall, memorize, and/or compare lengthy, alphanumeric identifiers by heuristically and deterministically mapping such identifiers to iconographic symbols based on a mathematical representation of the identifiers.

CONCLUSION

This disclosure describes techniques that map lengthy alphanumeric strings to easy-to-recall icons and/or colors. The map can be a deterministic, many-to-one function, e.g., a hashing function.

REFERENCES

- [1] https://en.wikipedia.org/wiki/X11_color_names accessed May 25, 2021.
- [2] <https://unicode.org/emoji/charts/full-emoji-list.html> accessed May 25, 2021.
- [3] <https://www.w3.org/TR/acname-1.1/> accessed May 25, 2021.