Proposal to guarantee stability of spelling of property names, values, and aliases in UCD

January, 2022 — Michael Ficarra, editor of ECMA-262

Proposed outcome

Stabilise the exact spellings (including case, whitspace, underscores, and hyphenation) of property names, values, and aliases in the Unicode Character Database to allow the use of strict matching within consuming specifications such as ECMA-262. The spellings chosen should be the first occurrence of each name, value, or alias in PropertyAliases.txt or PropertyValueAliases.txt, as appropriate. For the purposes of this proposal, abbreviated forms, long forms, and aliases are all distinct.

Background and Rationale

ECMA-262 is the standard that defines the ECMAScript (popularly known as JavaScript) programming language. TC39 (Technical Committee 39 of Ecma Intl) is the group of browser vendors and other web stakeholders that defines ECMA-262. ECMAScript currently allows the use of Unicode property names, values, and aliases within regular expressions, which may be embedded within an ECMAScript program. The following is an example of a regular expression which uses the feature to roughly match an ECMAScript identifier.

```
function isIdentifier(str) {
    return /^[\p{ID_Start}$_][\p{ID_Continue}$\u{200D}\u{200C}]*$/.test(str);
}
```

Note that, in the above example, `ID_Start` and `ID_Continue` must be written exactly as-is. Despite the following recommendation to use loose matching in PropertyAliases.txt and PropertyValueAliases.txt,

> ```
> # Loose matching should be applied to all property names and property values, with
> # the exception of String Property values. With loose matching of property names and
> # values, the case distinctions, whitespace, hyphens, and '_' are ignored.
> # For Numeric Property values, numeric equivalence is applied: thus "01.00"
> # is equivalent to "1".
> ```

TC39 has made (and recently reaffirmed) the decision to instead mandate strict matching, only allowing a single spelling for any property name, value, or alias. The allowed spellings can be seen in tables 67 (Non-binary Unicode property aliases and their canonical property names), 68 (Binary Unicode property aliases and their canonical property names), 69 (Value aliases and canonical values for the Unicode property General_Category), and 70 (Value aliases and canonical values for the Unicode properties Script and Script_Extensions) of ECMA-262. The former two tables must exist within ECMA-262 because TC39 does not want to automatically include new Unicode property names and aliases within ECMAScript upon each Unicode

release. However, TC39 and especially the editors of ECMA-262 would prefer to refer to the UCD instead of including the latter two tables. Unfortunately, because the UCD does not include canonical spellings of property names, values, or aliases, and because their spelling is neither consistent throughout nor guaranteed stable, we cannot make this reference.

ECMAScript is an extremely widely used programming language, and may already have defined a de facto canonical spelling. Developers who want to write maximally portable regular expressions will use the spellings mandated by ECMAScript. To maximise the portability of ECMAScript regular expressions to other languages which may also mandate strict matching of property names, values, and aliases, the UCD should provide canonical spellings.

The spellings defined in ECMA-262 were chosen to be the first spelling used in PropertyAliases.txt and PropertyValueAliases.txt at the time they were added. As of now, that continues to be the stated process of TC39 for inclusion of future properties. No spellings of property names, values, or aliases in ECMA-262 differ from the spellings in PropertyAliases.txt and PropertyValueAliases.txt. ECMA-262 includes the General_Category property values Any, Assigned, and ASCII, as recommended in UTS#18.

While the decision by TC39 for ECMAScript regular expressions to use strict matching was an elective design choice, there are other contexts where strict matching must be used due to external constraints. For example, UAX#44 states:

> Both Unicode character properties themselves and their values are given symbolic aliases. The formal lists of aliases are provided so that well-defined symbolic values are available for XML formats of the UCD data, for regular expression property tests, and for other programmatic textual descriptions of Unicode data.

However, XML element and attribute names are matched strictly. This means that, in order for Unicode aliases to be useful in "XML formats of the UCD data" as UAX #44 suggests, their spellings should be stable under strict matching.