Юникод: флаг "u" и класс \p{...}

В JavaScript для строк используется кодировка Юникод. Обычно символы кодируются с помощью 2 байтов, что позволяет закодировать максимум 65536 символов.

Этого диапазона не хватает для того, чтобы закодировать все символы. Поэтому некоторые редкие символы кодируются с помощью 4 байтов, например 𝒳 (математический X) или 😄 (смайлик), некоторые иероглифы, и т.п.

В таблице ниже приведены юникоды нескольких символов:

Символ	Юникод	Количество байт в юникоде
a	0x0061	2
≈	0x2248	2
𝒳	0x1d4b3	4
𝒴	0x1d4b4	4
😄	0x1f604	4

Таким образом, символы типа a и ≈ занимают по 2 байта, а коды для 𝒳, 𝒴 и 😄 – длиннее, в них 4 байта.

Когда-то давно, на момент создания языка JavaScript, кодировка Юникод была проще: символов в 4 байта не существовало. И, хотя это время давно прошло, многие строковые функции всё ещё могут работать некорректно.

Например, свойство `length` считает, что здесь два символа:

```
alert('😄'.length); // 2
alert('𝒳'.length); // 2
```

...Но мы видим, что только один, верно? Дело в том, что свойство `length` воспринимает 4-байтовый символ как два символа по 2 байта. Это неверно, потому что эти два символа должны восприниматься как единое целое (так называемая «суррогатная пара», вы также можете прочитать об этом в главе Строки).

Регулярные выражения также по умолчанию воспринимают 4-байтные «длинные символы» как пары 2-байтных. Как и со строками, это может приводить к странным результатам. Мы увидим примеры чуть позже, в главе Наборы и диапазоны [...].

В отличие от строк, у регулярных выражений есть специальный флаг `u`, который исправляет эту проблему. При его наличии регулярное выражение работает с 4-байтными символами правильно. И, кроме того, становится доступным поиск по юникодным свойствам, который мы рассмотрим далее.

Юникодные свойства \p{...}

Не поддерживается в некоторых старых браузерах

Несмотря на то, что это часть стандарта с 2018 года, юникодные свойства не поддерживаются в Firefox до 78 версии и в Edge до 79 версии.

Существует библиотека XRegExp, которая реализует «расширенные» регулярные выражения с кросс-браузерной поддержкой юникодных свойств.

Каждому символу в кодировке Юникод соответствует множество свойств. Они описывают к какой «категории» относится символ, содержат различную информацию о нём.

Например, свойство `Letter` у символа означает, что это буква какого-то алфавита, причём любого. А свойство `Number` означает, что это цифра – арабская или китайская, и т.п, на каком-то из языков.

В регулярном выражении можно искать символ с заданным свойством, указав его в `\p{…}`. Для таких регулярных выражений обязательно использовать флаг `u`.

Например, `\p{Letter}` обозначает букву в любом языке. Также можно использовать запись `\p{L}`, так как `L` – это псевдоним `Letter`. Существуют короткие записи почти для всех свойств.

В примере ниже будут найдены английская, грузинская и корейская буквы:

```
let str = "A ბ ㄱ";

alert( str.match(/\p{L}/gu) ); // A,ბ,ㄱ
alert( str.match(/\p{L}/g) ); // null (ничего не нашло,
```

Вот основные категории символов и их подкатегории:

- Буквы `L`:
  - в нижнем регистре `Ll`,
  - модификаторы `Lm`,
  - заглавные буквы `Lt`,
  - в верхнем регистре `Lu`,
  - прочие `Lo`.
- Числа `N`:
  - десятичная цифра `Nd`,
  - цифры обозначаемые буквами (римские) `Nl`,
  - прочие `No`.
- Знаки пунктуации `P`:
  - соединители `Pc`,
  - тире `Pd`,
  - открывающие кавычки `Pi`,
  - закрывающие кавычки `Pf`,
  - открывающие скобки `Ps`,
  - закрывающие скобки `Pe`,
  - прочее `Po`.
- Отметки `M` (например, акценты):
  - двоеточия `Mc`,
  - вложения `Me`,
  - апострофы `Mn`.
- Символы `S`:
  - валюты `Sc`, модификаторы `Sk`, математические `Sm`, прочие `So`.
- Разделители `Z`:
  - линия `Zl`,
  - параграф `Zp`,
  - пробел `Zs`.
- Прочие `C`:
  - контрольные `Cc`,
  - форматирование `Cf`,
  - не назначенные `Cn`,
  - для приватного использования `Co`,
  - суррогаты `Cs`.

Так что, например, если нам нужны буквы в нижнем регистре, то можно написать `\p{Ll}`, знаки пунктуации: `\p{P}` и так далее.

Есть и другие категории – производные, например:

- `Alphabetic` (`Alpha`), включающая в себя буквы `L`, плюс «буквенные цифры» `Nl` (например XII – символ для римской записи числа 12), и некоторые другие символы `Other_Alphabetic` (`OAlpha`).
- `Hex_Digit` включает символы для шестнадцатеричных чисел: `0-9`, `a-f`.
- И так далее.

Юникод поддерживает много различных свойств, их полное перечисление потребовало бы очень много места, поэтому вот ссылки:

- По символу посмотреть его свойства: https://unicode.org/cldr/utility/character.jsp.
- По свойству посмотреть символы с ним: https://unicode.org/cldr/utility/list-unicodeset.jsp.
- Короткие псевдонимы для свойств: https://www.unicode.org/Public/UCD/latest/ucd/PropertyValueAliases.txt.
- Полная база Юникод-символов в текстовом формате вместе со всеми свойствами, находится здесь: https://www.unicode.org/Public/UCD/latest/ucd/.

Пример: шестнадцатеричные числа

Например, давайте поищем шестнадцатеричные числа, записанные в формате `xFF`, где вместо `F` может быть любая шестнадцатеричная цифра (0…1 или A…F).

Шестнадцатеричная цифра может быть обозначена как `\p{Hex_Digit}`:

```
let regexp = /x\p{Hex_Digit}\p{Hex_Digit}/u;

alert("число: xAF".match(regexp)); // xAF
```

Пример: китайские иероглифы

Поищем китайские иероглифы.

В Юникоде есть свойство `Script` (система написания), которое может иметь значения `Cyrillic` (Кириллическая), `Greek` (Греческая), `Arabic` (Арабская), `Han` (Китайская) и так далее, здесь полный список.

Для поиска символов в нужной системе мы должны установить `Script=<значение>`, например для поиска кириллических букв: `\p{sc=Cyrillic}`, для китайских иероглифов: `\p{sc=Han}`, и так далее:

```
let regexp = /\p{sc=Han}/gu; // вернёт китайские иерогл

let str = `Hello Привет 你好 123_456`;

alert( str.match(regexp) ); // 你,好
```

Пример: валюта

Символы, обозначающие валюты, такие как `$`, `€`, `¥` и другие, имеют свойство `\p{Currency_Symbol}`, короткая запись `\p{Sc}`.

Используем его, чтобы поискать цены в формате «валюта, за которой идёт цифра»:

```
let regexp = /\p{Sc}\d/gu;

let  str = `Цены: $2, €1, ¥9`;

alert( str.match(regexp) ); // $2,€1,¥9
```

Позже, в главе Квантификаторы +, *, ? и {n} мы изучим, как искать числа из любого количества цифр.

Итого

Флаг `u` включает поддержку Юникода в регулярных выражениях.

Конкретно, это означает, что:

1. Символы из 4 байт воспринимаются как единое целое, а не как два символа по 2 байта.
2. Работает поиск по юникодным свойствам `\p{…}`.

С помощью юникодных свойств мы можем искать слова на нужных языках, специальные символы (кавычки, обозначения валюты) и так далее.

Комментарии

перед тем как писать...