import {Value} from './index';

/\*\*

\* Sass's [string type](https://sass-lang.com/documentation/values/strings).

\*

\* @category Custom Function

\*/

export class SassString extends Value {

/\*\*

\* Creates a new string.

\*

\* @param text - The contents of the string. For quoted strings, this is the

\* semantic content—any escape sequences that were been written in the source

\* text are resolved to their Unicode values. For unquoted strings, though,

\* escape sequences are preserved as literal backslashes.

\*

\* @param options.quotes - Whether the string is quoted. Defaults to `true`.

\*/

constructor(

text: string,

options?: {

quotes?: boolean;

}

);

/\*\*

\* Creates an empty string.

\*

\* @param options.quotes - Whether the string is quoted. Defaults to `true`.

\*/

constructor(options?: {quotes?: boolean});

/\*\*

\* The contents of the string.

\*

\* For quoted strings, this is the semantic content—any escape sequences that

\* were been written in the source text are resolved to their Unicode values.

\* For unquoted strings, though, escape sequences are preserved as literal

\* backslashes.

\*

\* This difference allows us to distinguish between identifiers with escapes,

\* such as `url\u28 http://example.com\u29`, and unquoted strings that contain

\* characters that aren't valid in identifiers, such as

\* `url(http://example.com)`. Unfortunately, it also means that we don't

\* consider `foo` and `f\6F\6F` the same string.

\*/

get text(): string;

/\*\* Whether this string has quotes. \*/

get hasQuotes(): boolean;

/\*\*

\* Sass's notion of this string's length.

\*

\* Sass treats strings as a series of Unicode code points while JavaScript

\* treats them as a series of UTF-16 code units. For example, the character

\* U+1F60A SMILING FACE WITH SMILING EYES is a single Unicode code point but

\* is represented in UTF-16 as two code units (`0xD83D` and `0xDE0A`). So in

\* JavaScript, `"n😊b".length` returns `4`, whereas in Sass

\* `string.length("n😊b")` returns `3`.

\*/

get sassLength(): number;

/\*\*

\* Converts `sassIndex` to a JavaScript index into [[text]].

\*

\* Sass indices are one-based, while JavaScript indices are zero-based. Sass

\* indices may also be negative in order to index from the end of the string.

\*

\* In addition, Sass indices refer to Unicode code points while JavaScript

\* string indices refer to UTF-16 code units. For example, the character

\* U+1F60A SMILING FACE WITH SMILING EYES is a single Unicode code point but

\* is represented in UTF-16 as two code units (`0xD83D` and `0xDE0A`). So in

\* JavaScript, `"n😊b".charCodeAt(1)` returns `0xD83D`, whereas in Sass

\* `string.slice("n😊b", 1, 1)` returns `"😊"`.

\*

\* This function converts Sass's code point indices to JavaScript's code unit

\* indices. This means it's O(n) in the length of `text`.

\*

\* @throws `Error` - If `sassIndex` isn't a number, if that number isn't an

\* integer, or if that integer isn't a valid index for this string.

\*/

sassIndexToStringIndex(sassIndex: Value, name?: string): number;

}