import {List, ValueObject} from 'immutable';

import {SassBoolean} from './boolean';

import {SassColor} from './color';

import {SassFunction} from './function';

import {ListSeparator} from './list';

import {SassMap} from './map';

import {SassNumber} from './number';

import {SassString} from './string';

export {SassArgumentList} from './argument\_list';

export {SassBoolean, sassTrue, sassFalse} from './boolean';

export {SassColor} from './color';

export {SassFunction} from './function';

export {SassList, ListSeparator} from './list';

export {SassMap} from './map';

export {SassNumber} from './number';

export {SassString} from './string';

/\*\*

\* Sass's [`null` value](https://sass-lang.com/documentation/values/null).

\*

\* @category Custom Function

\*/

export const sassNull: Value;

/\*\*

\* The abstract base class of Sass's value types.

\*

\* This is passed to and returned by [[CustomFunction]]s, which are passed into

\* the Sass implementation using [[Options.functions]].

\*

\* @category Custom Function

\*/

export abstract class Value implements ValueObject {

protected constructor();

/\*\*

\* This value as a list.

\*

\* All SassScript values can be used as lists. Maps count as lists of pairs,

\* and all other values count as single-value lists.

\*

\* @returns An immutable [[List]] from the [`immutable`

\* package](https://immutable-js.com/).

\*/

get asList(): List<Value>;

/\*\*

\* Whether this value as a list has brackets.

\*

\* All SassScript values can be used as lists. Maps count as lists of pairs,

\* and all other values count as single-value lists.

\*/

get hasBrackets(): boolean;

/\*\*

\* Whether the value counts as `true` in an `@if` statement and other

\* contexts.

\*/

get isTruthy(): boolean;

/\*\*

\* Returns JavaScript's `null` value if this is [[sassNull]], and returns

\* `this` otherwise.

\*/

get realNull(): null | Value;

/\*\*

\* The separator for this value as a list.

\*

\* All SassScript values can be used as lists. Maps count as lists of pairs,

\* and all other values count as single-value lists.

\*/

get separator(): ListSeparator;

/\*\*

\* Converts `sassIndex` into a JavaScript-style index into the list returned

\* by [[asList]].

\*

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

\* indexes may also be negative in order to index from the end of the list.

\*

\* @param sassIndex - The Sass-style index into this as a list.

\* @param name - The name of the function argument `sassIndex` came from

\* (without the `$`) if it came from an argument. Used for error reporting.

\* @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 [[asList]].

\*/

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

/\*\*

\* Returns the value at index `index` in this value as a list, or `undefined`

\* if `index` isn't valid for this list.

\*

\* All SassScript values can be used as lists. Maps count as lists of pairs,

\* and all other values count as single-value lists.

\*

\* This is a shorthand for `this.asList.get(index)`, although it may be more

\* efficient in some cases.

\*

\* \*\*Heads up!\*\* This method uses the same indexing conventions as the

\* `immutable` package: unlike Sass the index of the first element is 0, but

\* like Sass negative numbers index from the end of the list.

\*/

get(index: number): Value | undefined;

/\*\*

\* Throws if `this` isn't a [[SassBoolean]].

\*

\* \*\*Heads up!\*\* Functions should generally use [[isTruthy]] rather than

\* requiring a literal boolean.

\*

\* @param name - The name of the function argument `this` came from (without

\* the `$`) if it came from an argument. Used for error reporting.

\*/

assertBoolean(name?: string): SassBoolean;

/\*\*

\* Throws if `this` isn't a [[SassColor]].

\*

\* @param name - The name of the function argument `this` came from (without

\* the `$`) if it came from an argument. Used for error reporting.

\*/

assertColor(name?: string): SassColor;

/\*\*

\* Throws if `this` isn't a [[SassFunction]].

\*

\* @param name - The name of the function argument `this` came from (without

\* the `$`) if it came from an argument. Used for error reporting.

\*/

assertFunction(name?: string): SassFunction;

/\*\*

\* Throws if `this` isn't a [[SassMap]].

\*

\* @param name - The name of the function argument `this` came from (without

\* the `$`) if it came from an argument. Used for error reporting.

\*/

assertMap(name?: string): SassMap;

/\*\*

\* Throws if `this` isn't a [[SassNumber]].

\*

\* @param name - The name of the function argument `this` came from (without

\* the `$`) if it came from an argument. Used for error reporting.

\*/

assertNumber(name?: string): SassNumber;

/\*\*

\* Throws if `this` isn't a [[SassString]].

\*

\* @param name - The name of the function argument `this` came from (without

\* the `$`) if it came from an argument. Used for error reporting.

\*/

assertString(name?: string): SassString;

/\*\*

\* Returns `this` as a map if it counts as one (empty lists count as empty

\* maps) or `null` if it doesn't.

\*/

tryMap(): SassMap | null;

/\*\* Returns whether `this` represents the same value as `other`. \*/

equals(other: Value): boolean;

/\*\* Returns a hash code that can be used to store `this` in a hash map. \*/

hashCode(): number;

/\*\* @hidden \*/

toString(): string;

}