// Type definitions for webidl-conversions 6.1

// Project: https://github.com/jsdom/webidl-conversions#readme

// Definitions by: ExE Boss <https://github.com/ExE-Boss>

// Definitions: https://github.com/DefinitelyTyped/DefinitelyTyped

// TypeScript Version: 3.0

type Parameters<T extends (...args: any[]) => any> = T extends (...args: infer P) => any ? P : never;

declare namespace WebIDLConversions {

interface Globals {

[key: string]: any;

Number: (value?: any) => number;

String: (value?: any) => string;

TypeError: new (message?: string) => TypeError;

}

interface Options {

context?: string | undefined;

globals?: Globals | undefined;

}

interface IntegerOptions extends Options {

enforceRange?: boolean | undefined;

clamp?: boolean | undefined;

}

interface StringOptions extends Options {

treatNullAsEmptyString?: boolean | undefined;

}

interface BufferSourceOptions extends Options {

allowShared?: boolean | undefined;

}

type IntegerConversion = (V: any, opts?: IntegerOptions) => number;

type StringConversion = (V: any, opts?: StringOptions) => string;

type NumberConversion = (V: any, opts?: Options) => number;

}

declare const WebIDLConversions: {

any<V>(V: V, opts?: WebIDLConversions.Options): V;

void(V?: any, opts?: WebIDLConversions.Options): void;

boolean(V: any, opts?: WebIDLConversions.Options): boolean;

byte(V: any, opts?: WebIDLConversions.IntegerOptions): number;

octet(V: any, opts?: WebIDLConversions.IntegerOptions): number;

short(V: any, opts?: WebIDLConversions.IntegerOptions): number;

['unsigned short'](V: any, opts?: WebIDLConversions.IntegerOptions): number;

long(V: any, opts?: WebIDLConversions.IntegerOptions): number;

['unsigned long'](V: any, opts?: WebIDLConversions.IntegerOptions): number;

['long long'](V: any, opts?: WebIDLConversions.IntegerOptions): number;

['unsigned long long'](V: any, opts?: WebIDLConversions.IntegerOptions): number;

double(V: any, opts?: WebIDLConversions.Options): number;

['unrestricted double'](V: any, opts?: WebIDLConversions.Options): number;

float(V: any, opts?: WebIDLConversions.Options): number;

['unrestricted float'](V: any, opts?: WebIDLConversions.Options): number;

DOMString(V: any, opts?: WebIDLConversions.StringOptions): string;

ByteString(V: any, opts?: WebIDLConversions.StringOptions): string;

USVString(V: any, opts?: WebIDLConversions.StringOptions): string;

object<V>(V: V, opts?: WebIDLConversions.Options): V extends object ? V : V & object;

ArrayBuffer(V: any, opts?: WebIDLConversions.BufferSourceOptions & { allowShared?: false | undefined }): ArrayBuffer;

ArrayBuffer(V: any, opts?: WebIDLConversions.BufferSourceOptions): ArrayBufferLike;

DataView(V: any, opts?: WebIDLConversions.BufferSourceOptions): DataView;

Int8Array(V: any, opts?: WebIDLConversions.BufferSourceOptions): Int8Array;

Int16Array(V: any, opts?: WebIDLConversions.BufferSourceOptions): Int16Array;

Int32Array(V: any, opts?: WebIDLConversions.BufferSourceOptions): Int32Array;

Uint8Array(V: any, opts?: WebIDLConversions.BufferSourceOptions): Uint8Array;

Uint16Array(V: any, opts?: WebIDLConversions.BufferSourceOptions): Uint16Array;

Uint32Array(V: any, opts?: WebIDLConversions.BufferSourceOptions): Uint32Array;

Uint8ClampedArray(V: any, opts?: WebIDLConversions.BufferSourceOptions): Uint8ClampedArray;

Float32Array(V: any, opts?: WebIDLConversions.BufferSourceOptions): Float32Array;

Float64Array(V: any, opts?: WebIDLConversions.BufferSourceOptions): Float64Array;

ArrayBufferView(V: any, opts?: WebIDLConversions.BufferSourceOptions): ArrayBufferView;

BufferSource(V: any, opts?: WebIDLConversions.BufferSourceOptions & { allowShared?: false | undefined }): ArrayBuffer | ArrayBufferView;

BufferSource(V: any, opts?: WebIDLConversions.BufferSourceOptions): ArrayBufferLike | ArrayBufferView;

DOMTimeStamp(V: any, opts?: WebIDLConversions.Options): number;

// tslint:disable:ban-types

/\*\* @deprecated Will be removed in v7.0 \*/

Function<V>(V: V, opts?: WebIDLConversions.Options): V extends (...args: any[]) => any ? V : Function;

/\*\* @deprecated Will be removed in v7.0 \*/

VoidFunction<V>(

V: V,

opts?: WebIDLConversions.Options,

): V extends (...args: any[]) => any ? (...args: Parameters<V>) => void : Function;

};

// This can't use ES6 style exports, as those can't have spaces in export names.

export = WebIDLConversions;