/// <reference types="node"/>

/\*\* https://url.spec.whatwg.org/#url-representation \*/

export interface URLRecord {

scheme: string;

username: string;

password: string;

host: string | number | IPv6Address | null;

port: number | null;

path: string[];

query: string | null;

fragment: string | null;

cannotBeABaseURL?: boolean;

}

/\*\* https://url.spec.whatwg.org/#concept-ipv6 \*/

export type IPv6Address = [number, number, number, number, number, number, number, number];

/\*\* https://url.spec.whatwg.org/#url-class \*/

export class URL {

constructor(url: string, base?: string | URL);

get href(): string;

set href(V: string);

get origin(): string;

get protocol(): string;

set protocol(V: string);

get username(): string;

set username(V: string);

get password(): string;

set password(V: string);

get host(): string;

set host(V: string);

get hostname(): string;

set hostname(V: string);

get port(): string;

set port(V: string);

get pathname(): string;

set pathname(V: string);

get search(): string;

set search(V: string);

get searchParams(): URLSearchParams;

get hash(): string;

set hash(V: string);

toJSON(): string;

readonly [Symbol.toStringTag]: "URL";

}

/\*\* https://url.spec.whatwg.org/#interface-urlsearchparams \*/

export class URLSearchParams {

constructor(

init?:

| ReadonlyArray<readonly [string, string]>

| Iterable<readonly [string, string]>

| { readonly [name: string]: string }

| string,

);

append(name: string, value: string): void;

delete(name: string): void;

get(name: string): string | null;

getAll(name: string): string[];

has(name: string): boolean;

set(name: string, value: string): void;

sort(): void;

keys(): IterableIterator<string>;

values(): IterableIterator<string>;

entries(): IterableIterator<[string, string]>;

forEach<THIS\_ARG = void>(

callback: (this: THIS\_ARG, value: string, name: string, searchParams: this) => void,

thisArg?: THIS\_ARG,

): void;

readonly [Symbol.toStringTag]: "URLSearchParams";

[Symbol.iterator](): IterableIterator<[string, string]>;

}

/\*\* https://url.spec.whatwg.org/#concept-url-parser \*/

export function parseURL(

input: string,

options?: { readonly baseURL?: string; readonly encodingOverride?: string },

): URLRecord | null;

/\*\* https://url.spec.whatwg.org/#concept-basic-url-parser \*/

export function basicURLParse(

input: string,

options?: {

baseURL?: string;

encodingOverride?: string;

url?: URLRecord;

stateOverride?: StateOverride;

},

): URLRecord | null;

/\*\* https://url.spec.whatwg.org/#scheme-start-state \*/

export type StateOverride =

| "scheme start"

| "scheme"

| "no scheme"

| "special relative or authority"

| "path or authority"

| "relative"

| "relative slash"

| "special authority slashes"

| "special authority ignore slashes"

| "authority"

| "host"

| "hostname"

| "port"

| "file"

| "file slash"

| "file host"

| "path start"

| "path"

| "cannot-be-a-base-URL path"

| "query"

| "fragment";

/\*\* https://url.spec.whatwg.org/#concept-url-serializer \*/

export function serializeURL(urlRecord: URLRecord, excludeFragment?: boolean): string;

/\*\* https://url.spec.whatwg.org/#concept-host-serializer \*/

export function serializeHost(host: string | number | IPv6Address): string;

/\*\* https://url.spec.whatwg.org/#serialize-an-integer \*/

export function serializeInteger(number: number): string;

/\*\* https://html.spec.whatwg.org#ascii-serialisation-of-an-origin \*/

export function serializeURLOrigin(urlRecord: URLRecord): string;

/\*\* https://url.spec.whatwg.org/#set-the-username \*/

export function setTheUsername(urlRecord: URLRecord, username: string): void;

/\*\* https://url.spec.whatwg.org/#set-the-password \*/

export function setThePassword(urlRecord: URLRecord, password: string): void;

/\*\* https://url.spec.whatwg.org/#cannot-have-a-username-password-port \*/

export function cannotHaveAUsernamePasswordPort(urlRecord: URLRecord): boolean;

/\*\* https://url.spec.whatwg.org/#percent-decode \*/

export function percentDecode(buffer: Extract<NodeJS.TypedArray, ArrayLike<number>>): Buffer;